

Configure DNS and ENUM

OPERATING INSTRUCTIONS

Copyright

© Ericsson AB 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

| | | |
|----------|-----------------------------------|------------|
| 1 | Introduction | 1 |
| 1.1 | Prerequisites | 1 |
| 1.2 | Relation Information | 2 |
| 2 | Overview | 3 |
| 3 | Configuring DNS | 5 |
| 3.1 | Configuring DNS Basic SW | 5 |
| 3.2 | Configuring ASDNS | 32 |
| 3.3 | DNS Server IPv6 Support | 40 |
| 3.4 | DNS Operation | 41 |
| 3.5 | ASDNS Operation | 58 |
| 4 | Configuring ENUM | 67 |
| 4.1 | Configuring ENUM | 67 |
| 4.2 | Configuring ENUM Access Control | 89 |
| 4.3 | Configuring ENUM Query Preference | 91 |
| 4.4 | Configuring Number Portability | 94 |
| 4.5 | Configuring RCSe Interconnect | 113 |
| 4.6 | Configuring Split Namespace | 115 |
| 4.7 | Configuring ENUM Front End | 119 |
| 4.8 | ENUM Server IPv6 Support | 129 |
| 4.9 | ENUM Operations | 130 |
| | Reference List | 133 |





1 Introduction

This document describes how to use the Configuration Management (CM) function in the IPWorks DNS and ENUM.

Target Groups

This document is intended for personnel configuring and fine-tuning IPWorks DNS and ENUM.

1.1 Prerequisites

This section states the prerequisites that must be fulfilled.

- Intermediate Linux and UNIX skills
- Concepts, terminologies, and telecommunication abbreviations, such as TCP/IP, packet data networks, and protocol servers
- An Ericsson Command-Line Interface (ECLI) session in Exec mode is in progress.

1.1.1 Documents

Before starting this procedure, ensure that the following web site and documents are available:

- For more information about the basics and concepts regarding the configuration management of IPWorks, refer to *IPWorks Configuration Management*.
- For more information about the objects configured through IPWorks CLI (ipwcli), refer to *IPWorks DNS, ASDNS, ENUM Parameter Description*.
- For more information about the objects configured through ECLI, refer to *Managed Object Model (MOM)*.
- For more information about how to use the IPWorks CLI, refer to *Command Line Interface User Guide for IPWorks SS*.

1.1.2 Tools

Not applicable.



1.1.3 Conditions

Before starting this procedure, the following conditions must apply:

- IPWorks installation is completed.
- Each functionality, associated with a specific license, must be valid and running normally in the license server.

For more information about IPWorks license related information, refer to *License Management*.

- Storage Server is started.
- Each server (such as DNS and ENUM) must be initially configured.

1.2 Relation Information

Trademark information, typographic conventions, and definition and explanation of abbreviations and terminology can be found in the following documents:

- *Trademark Information*
- *Typographic Conventions*
- *Glossary of Terms and Acronyms*



2 Overview

Table 1 Configuration Overview

| Functionality | | Procedure |
|---------------|--------------------------|-----------------------------|
| DNS | DNS Basic SW | See Section 3.1 on page 5 |
| | ASDNS | See Section 3.2 on page 32 |
| | DNS Server IPv6 Support | See Section 3.3 on page 40 |
| ENUM | ENUM | See Section 4.1 on page 67 |
| | ENUM Access Control | See Section 4.2 on page 88 |
| | ENUM Query Preference | See Section 4.3 on page 91 |
| | Number Portability | See Section 4.4 on page 94 |
| | RCSe Interconnect | See Section 4.5 on page 113 |
| | Split Namespace | See Section 4.6 on page 115 |
| | ENUM Front End | See Section 4.7 on page 119 |
| | ENUM Server IPv6 Support | See Section 4.8 on page 129 |





3 Configuring DNS

This section guides the configuration personnel how to configure DNS in IPWorks.

The IPWorks DNS Management can be used for DNS configuration. For more information, refer to *IPWorks DNS Management User Guide*.

3.1 Configuring DNS Basic SW

This section provides the sample configuration of DNS Basic SW in the sample networks.

Prerequisites:

Before configuring DNS, ensure that the Storage Server is started. If the Storage Server is not started, execute the following shell commands:

1. Log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>
```

2. Start the Storage Server on SC.

```
#ipw-ctr start ss SC-1  
start ss ==> success.
```

```
#ipw-ctr start ss SC-2  
start ss ==> success.
```

Actions:

The configuration of DNS Basic SW contains the following topics:

- Section 3.1.1 Configuring DNS in Sample Network 1 on page 5
- Section 3.1.2 Configuring DNS in Sample Network 2 on page 12
- Section 3.1.3 Configuring Resource Records on page 18.
- Section 3.1.4 Configuration of Zones on page 23.
- Section 3.1.5 Configuring DNS Caching on page 30.

3.1.1 Configuring DNS in Sample Network 1

The example in this section is based on the configuration of the sample GPRS network as shown in Figure 1.

Note: In *Sample Network 1*, the internal DNS (iDNS) and external DNS (eDNS) are deployed on two separated IPWorks systems.

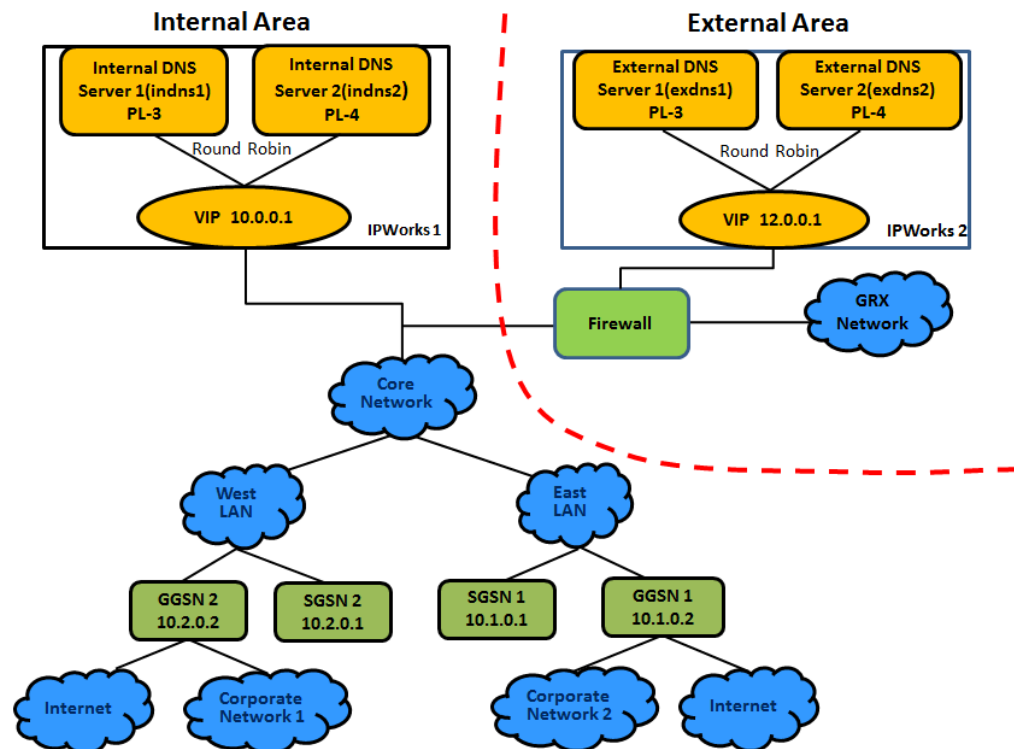


Figure 1 Sample Network 1

The *Sample Network 1* has the following characteristics:

- It is assumed that the network infrastructure systems are on a network called `example.net`.
- The network is split into two networks. The internal network is only accessible from other internal systems or through a firewall that also performs address translation. The external network handles access from external hosts.
- IPWorks system in internal/external network is one cluster environment. IPWorks system configuration is 2 SCs + 2 PLs. For each cluster environment, the DNS servers are located on 2 PL blades and the traffic is shared by them through round robin mechanism.
- The internal and external network each has two master DNS servers.
- The internal network uses addresses in the `10.0.0.0/8` network, whereas the external network uses addresses in the `12.0.0.0/8` network.
- The *Sample Network 1* is divided into two geographical regions, which are called `east` and `west`. DNS queries from hosts in the `east` region prefer the GGSNs from the `east` region and queries from the `west` prefer



GGSNs from the *west*. Each regional LAN acts as a backup for the other if one of the GGSNs becomes unavailable. The *ActiveSelect* feature of the IPWorks DNS servers can be used to implement this.

- The *Sample Network 1* is assumed to be in the *active partition*. For more information about the partition, refer to the *Partition* section in *IPWorks Configuration Management*.

To configure DNS in *Sample Network 1*, perform the following actions:

- Section 3.1.1.1 Initializing IPWorks for Sample Network 1 on page 7
- Section 3.1.1.2 Configuring DNS Servers for Sample Network 1 on page 8
- Section 3.1.1.3 Using Default View for Sample Network 1 on page 10
- Section 3.1.1.4 Configuring Master Zones for Sample Network 1 on page 10
- Section 3.1.1.5 Configuring DNS Resource Records for Sample Network 1 on page 11

3.1.1.1

Initializing IPWorks for Sample Network 1

In the *Sample Network 1*, two separate *areas* are used for the internal and the external networks respectively. These are called *internalarea* and *externalarea* to keep the naming scheme simple.

For more information on *area*, refer to the *Area* section in *IPWorks Configuration Management*.

Note: In this document:

- For the IPWorks inside the internal network, it is named *internal IPWorks*.
- For the IPWorks inside the external network, it is named *external IPWorks*.

In internal IPWorks:

```
IPWorks> create area internalarea -set description="Area for internal data"
1 object(s) created.
```

In external IPWorks:

```
IPWorks> create area externalarea -set description="Area for external data"
1 object(s) created.
```

To keep this example simple, the management of *PTRRecord* is not considered. For more information on PTR management, refer to the *PTRRecord* section in *IPWorks DNS, ASDNS, ENUM Parameter Description*. The following command disables any query that might arise during this session regarding *PTRRecord*.



```
IPWorks> set Auto.CreatePTRRecord no
```

Note: Never create *dnsserver* in IPWCLI for the scaled-out blade.

3.1.1.2 Configuring DNS Servers for Sample Network 1

In *Sample Network 1*, the internal and external networks, each has two master *DnsServers*. The cluster internal IP address on PLs is used as the address of *DnsServer* object in IPWorks CLI. The client uses the traffic VIP address to query data and the load balance is applied on PLs through round robin mechanism by VIP.

The cluster internal IP address on PLs can be got by following method (The `inet addr` under interface 'bond0'):

For example:

```
PL-3:~ # ifconfig bond0
eth0      Link encap:Ethernet  HWaddr 02:10:20:01:03:00
          inet addr:169.254.100.3  Bcast:169.254.100.255  Mask:255.255.255.0
          inet6 addr: fe80::10:20ff:fe01:300/64 Scope:Link
          inet6 addr: fd00::3/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1605548669 errors:0 dropped:16 overruns:0 frame:0
          TX packets:1628421941 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:255221617210 (243398.3 Mb)  TX bytes:245805277728 (234418.1 Mb)
```

In internal IPWorks:

```
IPWorks> create dnsserver indns1 -set dnsname=indns1.example.net
t -set address=169.254.101.3
A default view is associated with this DNS server.
Which area will be used as the default area for zones in the "default" view?
This view is associated with DnsServer(s)[indns1]. The areas that exist
are[default,internalarea]. > internalarea
1 object(s) created.
```

```
IPWorks> create dnsserver indns2 -set dnsname=indns2.example.net
t -set address=169.254.101.4
A default view is associated with this DNS server.
Which area will be used as the default area for zones in the "default" view?
This view is associated with DnsServer(s)[indns2]. The areas that exist
are[default, internalarea]. > internalarea
1 object(s) created.
```

In external IPWorks:



```
IPWorks> create dnsserver exdns1 -set dnsname=exdns1.example.net
t -set address=169.254.101.3
A default view is associated with this DNS server.
Which area will be used as the default area for zones in the "default" view?
This view is associated with DnsServer(s) [exdns1]. The areas that exist
are [default,externalarea].> externalarea
1 object(s) created.

IPWorks> create dnsserver exdns2 -set dnsname=exdns2.example.net
t -set address=169.254.101.4
A default view is associated with this DNS server.
Which area will be used as the default area for zones in the "default" view?
This view is associated with DnsServer(s) [exdns2]. The areas that exist
are [default,externalarea].> externalarea
1 object(s) created.
```

Start the corresponding DNS server and DNS Server Manager, and update it through IPWorks CLI in both internal and external network.

In internal IPWorks:

```
# ipw-ctr start dnssm <PL hostname>
# ipw-ctr start dns <PL hostname>
IPWorks> update dnsserver
IPWorks> list dnsserver
[DnsServer indns2]
  Partition: active
  Name: indns2
  Address: 169.254.101.4
  PrimaryAddress: 169.254.101.4
  DnsName: indns2.example.net
  PrimaryDnsName: indns2.example.net
  Filename: named.conf
  AlgServerType: false
  ExportNeeded: true
[DnsServer indns1]
  Partition: active
  Name: indns1
  Address: 169.254.101.3
  PrimaryAddress: 169.254.101.3
  DnsName: indns1.example.net
  PrimaryDnsName: indns1.example.net
  Filename: named.conf
  AlgServerType: false
  ExportNeeded: true
```

In external IPWorks:

```
# ipw-ctr start dnssm <PL hostname>
# ipw-ctr start dns <PL hostname>
IPWorks> update dnsserver
```

Note:

- When the same DNS server is recreated, use the command `update dnsserver` to update the DNS before creating the first zone. It helps to clean up the remained zone files to avoid conflict.
- Before executing the command `update dnsserver`, ensure that the DNS server on payload (PL) has been started. Otherwise, the command fails.



3.1.1.3 Using Default View for Sample Network 1

This example does not use *view* to implement a split namespace in the *Sample Network 1*. Thus, it is unnecessary to define any view within any of the servers. The default view is automatically created when a DNS server is created. Since there are multiple areas (default area, internal area or external area) and only a single view, IPWorks prompts:

Which area will be used as the default area for zones in the "default" view?

3.1.1.4 Configuring Master Zones for Sample Network 1

Each *MasterZone* object is associated with the same area. For the two DNS servers in both internal and external network, a *MasterZone* in each server with the same name is created. This means that any time a record is added to the area, it will be automatically added to each of the zones. Then, the redundancy can be provided and the traffic load can be shared by the two DNS servers.

Sample Network 1 has two GPRS zones with the same name, called *mnc001.mcc001.gprs* in internal network. This *MasterZone* represents the configuration of a single GPRS network operator mnc001 (mobile network code 001) in a single country mcc001 (mobile country code 001). This example does not use any dynamic zone.

In internal IPWorks:

```
IPWorks>create masterzone indns1 mnc001.mcc001.gprs -set area=internalarea
1 object(s) created.
IPWorks>list
[MasterZone mnc001.mcc001.gprs]
  Partition: active
  Server: indns1
  View: default
  Name: mnc001.mcc001.gprs
  ZoneId: indns1:default:mnc001.mcc001.gprs
  ZoneType: master
  Filename: db.mnc001.mcc001.gprs
  DefaultTtl: 86400
  ExportNeeded: true
  SourceName: indns1.example.net
  AuthoritativeName: indns1.example.net
  MasterZoneType: File
  IsDynamic: false

IPWorks> create masterzone indns2 mnc001.mcc001.gprs -set area=internalarea
1 object(s) created.
```

When the master zone is created, both the name of the zone and the server are specified. Thus, the *example.net* network infrastructure is also created.

In internal IPWorks:



```
IPWorks> create masterzone indns1 example.net -set area=internalarea
1 object(s) created.

IPWorks> create masterzone indns2 example.net -set area=internalarea
1 object(s) created.
```

In external IPWorks:

```
IPWorks> create masterzone exdns1 example.net -set area=externalarea
1 object(s) created.

IPWorks> create masterzone exdns2 example.net -set area=externalarea
1 object(s) created.
```

3.1.1.5

Configuring DNS Resource Records for Sample Network 1

It is important to remember that it is unnecessary to specify a value for the `container` field. IPWorks either computes this, or it prompts for a value if it cannot be uniquely determined.

The `container` also can be specified while creating the resource records:

In internal IPWorks:

```
IPWorks> create arecord internalarea indns1.example.net 10.0.0.11
1 object(s) created.
IPWorks> create arecord internalarea indns2.example.net 10.0.0.12
1 object(s) created.
```

In external IPWorks:

```
IPWorks> create arecord externalarea exdns1.example.net 12.0.0.11
1 object(s) created.
IPWorks> create arecord externalarea exdns2.example.net 12.0.0.12
1 object(s) created.
```

The previous examples illustrate how to create the *ARRecord* objects for the DNS servers. ARecords for GGSN and SGSN nodes can also be created:

Example in internal IPWorks:

```
IPWorks> create arecord internalarea sgsn1.example.net 10.1.0.1
1 object(s) created.
IPWorks> create arecord internalarea ggsn1.example.net 10.1.0.2
1 object(s) created.
IPWorks> create arecord internalarea sgsn2.example.net 10.2.0.1
1 object(s) created.
IPWorks> create arecord internalarea ggsn2.example.net 10.2.0.2
1 object(s) created.
```

To define an APN in the GPRS network, ARecords that bind the APN name to the address for the relevant GGSN nodes are created:



Example in internal IPWorks:

```
IPWorks> create arecord internalarea corp1.mnc001.mcc001.gprs 10.2.0.2
1 object(s) created.
IPWorks> create arecord internalarea corp2.mnc001.mcc001.gprs 10.1.0.2
1 object(s) created.
IPWorks> create arecord internalarea Internet.mnc001.mcc001.gprs 10.2.0.2
1 object(s) created.
IPWorks> create arecord internalarea Internet.mnc001.mcc001.gprs 10.1.0.2
1 object(s) created.
IPWorks> create arecord internalarea www.mnc001.mcc001.gprs 10.2.0.2
1 object(s) created.
IPWorks> create arecord internalarea www.mnc001.mcc001.gprs 10.1.0.2
1 object(s) created.
```

In the *Sample Network 1*, the default *NSRecord* and *SOARecord* created by IPWorks are used. Here is an example that displays the *NSRecord* objects (in their configuration file format). It is difficult to interpret the meanings of these records without knowing which zones they belong to.

In external IPWorks:

```
IPWorks> list nsrecords -format=conf
example.net.      IN NS exdns1.example.net.
example.net.      IN NS exdns2.example.net.
```

In internal IPWorks:

```
IPWorks> list nsrecords -format=conf
mnc001.mcc001.gprs. IN NS indns1.example.net.
mnc001.mcc001.gprs. IN NS indns2.example.net.
example.net.        IN NS indns1.example.net.
example.net.        IN NS indns2.example.net.
```

3.1.2 Configuring DNS in Sample Network 2

This section provides a sample configuration based on *Sample Network 2* (see Figure 2). Other than *Sample Network 1* that shows the DNS deployed on two separated IPWorks systems, *Sample Network 2* shows the Internal DNS (iDNS) and external DNS (eDNS) that are deployed on the same IPWorks system.

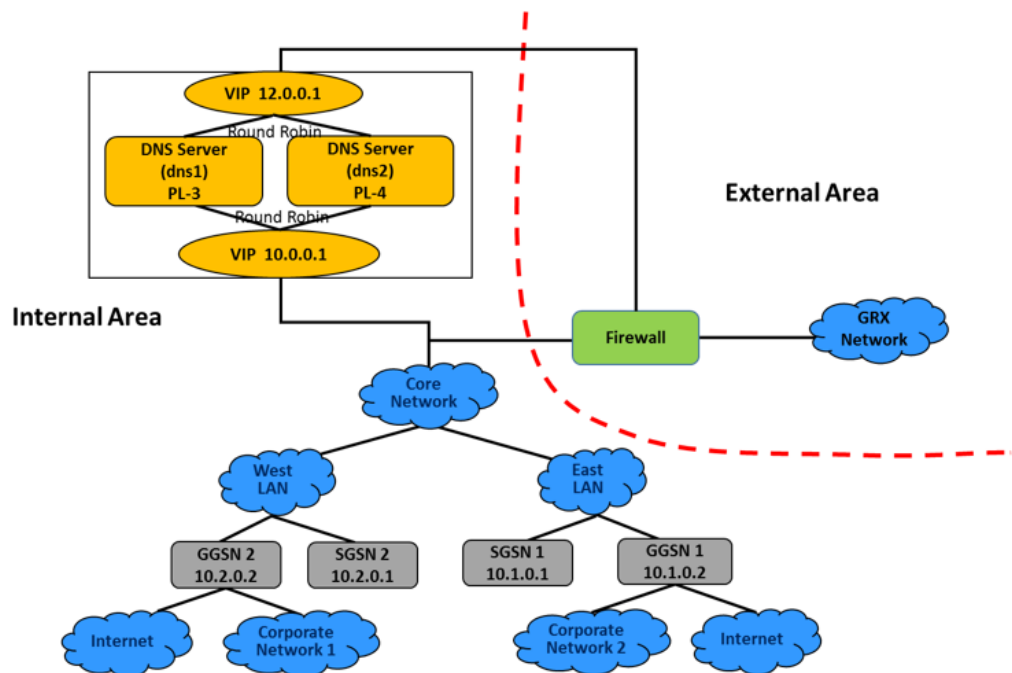


Figure 2 Sample Network 2

The *Sample Network 2* is based on the following conditions:

- IPWorks system is one cluster environment. IPWorks system deployment scenario is 2 SCs + 2 PLs. The DNS servers are located on 2 PLs and the traffic is shared by them through round robin mechanism.
- This network includes two master DNS servers and these servers handle the DNS traffic for both the internal and external networks.
- Both servers rely on views to implement the split namespace.
- One extra eVIP IP is added for the external network. (The external interface still can be protected by the firewall.)

Note: It is important that appropriate measures are taken to secure the system. A successful attack on these servers from an external source can impact DNS service to internal users as well as compromise access to the internal network.

Prerequisites:

Before the configuration, ensure that the Storage Server is started. If the Storage Server is not started, execute the following shell commands:

1. Log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>
```
2. Start the Storage Server on SC.



```
#ipw-ctr start ss SC-1
start ss ==> success.

#ipw-ctr start ss SC-2
start ss ==> success.
```

Actions:

The configuration contains the following topics:

- Section 3.1.2.1 Configuring eVIP IP for the External Network on page 14
- Section 3.1.2.2 Initializing IPWorks for Sample Network 2 on page 14
- Section 3.1.2.3 Configuring DNS Servers for Sample Network 2 on page 15
- Section 3.1.2.4 Configuring Named ACLs on page 16
- Section 3.1.2.5 Configuring Views for Sample Network 2 on page 16
- Section 3.1.2.6 Configuring Zones for Sample Network 2 on page 17
- Section 3.1.2.7 Configuring DNS Resource Records for Sample Network 2 on page 18

3.1.2.1 Configuring eVIP IP for the External Network

One extra eVIP IP must be added for the external network. Refer to *Add eVIP Traffic IP for IPWorks* for details.

3.1.2.2 Initializing IPWorks for Sample Network 2

In the example, two separate areas are used for the internal and the external networks respectively. These are called `internalarea` and `externalarea` to keep the naming scheme simple.

```
IPWorks> create area internalarea -set description="Area
for internal data"
```

```
1 object(s) created.
```

```
IPWorks> create area externalarea -set description="Area
for external data"
```

```
1 object(s) created.
```

To keep this example simple, the management of PTRRecords is not considered. The following command will disable any query that may arise during this session regarding PTRRecords.

```
IPWorks> set Auto.CreatePTRRecord no
```



3.1.2.3

Configuring DNS Servers for Sample Network 2

1. Fetch the cluster internal IP address (The `inet addr` under interface `eth0`) on PLs.

For example:

```
PL-3:~ # ifconfig eth0
```

```
eth0      Link encap:Ethernet  HWaddr 02:10:20:01:03:00
          inet addr:169.254.100.3  Bcast:169.254.100.255  Mask:255.255.255.0
          inet6 addr: fe80::10:20ff:fe01:300/64  Scope:Link
          inet6 addr: fd00::3/64  Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1605548669  errors:0  dropped:16  overruns:0  frame:0
          TX packets:1628421941  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:255221617210 (243398.3 Mb)  TX bytes:245805277728 (234418.1 Mb)
```

This IP address on PLs is used to create `DnsServer` object in Step 2.

2. Create DNS servers and associate them with related areas by using IPWorks CLI.

Note: In this example, `internalarea` is used.

```
IPWorks> create dnsserver dns1 -set dnsname=dns1.example.net -set address=169.254.100.3
```

```
A default view is asociated with this DNS server.
Which area will be used as the default area for
Zones in the "default" view? This view is associated
with DnsServer(s) [dns1]. The areas that exist are
[default,externalarea,internalarea]. [default] >
internalarea
```

```
1 object(s) created.
```

```
IPWorks> create dnsserver dns2 -set dnsname=dns2.example.net -set address=169.254.100.4
```

```
A default view is asociated with this DNS server.
Which area will be used as the default area for
Zones in the "default" view? This view is associated
with DnsServer(s) [dns2]. The areas that exist are
[default,externalarea,internalarea]. [default] >
internalarea
```

3. Verify the information of the DNS servers.

```
IPWorks> list dnsserver
```

```
[DnsServer dns1]
Partition: active
Name: dns1
```



```
Address: 169.254.100.3
PrimaryAddress: 169.254.100.3
DnsName: dns1.example.net
PrimaryDnsName: dns1.example.net
Filename: named.conf
AlgServerType: false
ExportNeeded: true
```

Note: The output of the list command is truncated to display only one DNS Server object. However, it will display all the relevant DNS servers (in this case, dns1 and dns2).

3.1.2.4 Configuring Named ACLs

When the views are defined, the match-clients option is also defined to determine which clients use the internal view and which clients use the external view. It is important that the two master servers use the same definition. Therefore a named ACL is used. It is assumed that any client in the 10.0.0.0/8 network is an internal client.

```
IPWorks> create acl internal-clients -set matchlist="{
10/8; }"
```

```
1 object(s) created.
```

3.1.2.5 Configuring Views for Sample Network 2

In this example, the namespace is split into two areas (internal and external) with two views in each server accordingly. The internal views have a match-clients option so that only internal clients can access the zones in these views. The external views do not have a match-clients option, which means that these views are accessible to all clients. In order to distinguish external clients from internal clients, the rank of internal views is set less than that of external views. Thereby all clients are first identified by internal views to see if they are internal.

```
IPWorks> create view dns1 internalview -set rank=1
-set area=internalarea -set option="match-clients {
internal-clients; }"
```

```
1 object(s) created.
```

```
IPWorks> create view dns2 internalview -set rank=1
-set area=internalarea -set option="match-clients {
internal-clients; }"
```

```
1 object(s) created.
```



```
IPWorks> create view dns1 externalview -set rank=2 -set  
area=externalarea
```

```
1 object(s) created.
```

```
IPWorks> create view dns2 externalview -set rank=2 -set  
area=externalarea
```

```
1 object(s) created.
```

3.1.2.6 Configuring Zones for Sample Network 2

In this example, zones are configured in the views by adding the name of the view as a key value between the name of the server and the name of the zone. If this is omitted, the zone is inserted into the default view.

```
IPWorks> create masterzone dns1 internalview mnc001.mcc00  
1.gprs
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns1 internalview example.net
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns2 internalview mnc001.mcc00  
1.gprs
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns2 internalview example.net
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns1 externalview mnc001.mcc00  
1.gprs
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns1 externalview example.net
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns2 externalview mnc001.mcc00  
1.gprs
```

```
1 object(s) created.
```

```
IPWorks> create masterzone dns2 externalview example.net
```



```
1 object(s) created.
```

3.1.2.7 Configuring DNS Resource Records for Sample Network 2

The ARecords for the servers, the SGSN and GGSN nodes are very similar to those defined in Section 3.1.1.5 Configuring DNS Resource Records for Sample Network 1 on page 11.

```
IPWorks> create arecord internalarea indns.example.net  
10.0.0.1
```

```
1 object(s) created.
```

```
IPWorks> create arecord externalarea exdns.example.net  
12.0.0.1
```

```
1 object(s) created.
```

```
IPWorks> create arecord internalarea sgsn1.example.net  
10.1.0.1
```

```
1 object(s) created.
```

```
IPWorks> create arecord internalarea ggsn1.example.net  
10.1.0.2
```

```
1 object(s) created.
```

```
IPWorks> create arecord internalarea sgsn2.example.net  
10.2.0.1
```

```
1 object(s) created.
```

```
IPWorks> create arecord internalarea ggsn2.example.net  
10.2.0.2
```

```
1 object(s) created.
```

3.1.3 Configuring Resource Records

This section describes configuration of certain resource records that are not used in any of the sample networks.

The configuration contains the following topics:

- Section 3.1.3.1 Using Subnets to Manage PTRRecords on page 19
- Section 3.1.3.2 Creating GenRecords on page 20



- Section 3.1.3.3 Implementing Classless in-addr.arpa Delegation on page 20
- Section 3.1.3.4 Implementing \$INCLUDE Directive on page 22

3.1.3.1

Using Subnets to Manage PTRRecords

To help clarify how the subnet-based PTR policies work, here is an example based on *Sample Network 1*.

In the *Sample Network 1*, four different subnets are defined. They represent the entire internal network, the West LAN, the East LAN, and the external network. The East LAN and West LAN are subsets of the overall Internal network (IPWorks allows for the definition of hierarchical subnets). The East and West LAN's (that will contain APNs) will not have *PTRRecord* created automatically. The rest of the Internal network will give a prompt about what should be done and the External network will always have *PTRRecord* created automatically. To configure this policy, the following objects must be created:

In internal IPWorks:

```
IPWorks> create subnet internalarea 10.0.0.0/8 -set ptrstrategy=prompt
-set description="Internal"
1 object(s) created.
IPWorks> create subnet internalarea 10.1.0.0/16 -set ptrstrategy=manual -set description="East LAN"
1 object(s) created.
IPWorks> create subnet internalarea 10.2.0.0/16 -set ptrstrategy=manual -set description="West LAN"
```

In external IPWorks:

```
1 object(s) created.
IPWorks> create subnet externalarea 12.0.0.0/8 -set ptrstrategy=auto -set description="External Network"
1 object(s) created.
IPWorks> list
[Subnet 12.0.0.0/8]
Partition: active
Area: default
Name: 12.0.0.0/8
Address: 12.0.0.0
Mask: 255.0.0.0
MaskLength: 8
FirstSortKey: 201326592
LastSortKey: 218103807
BroadcastAddress: 12.255.255.255
PtrStrategy: auto
IsDistinct: true
Server: none
Description: External
```

Now that the appropriate subnets have been created with some *ARecord* objects to see what the behavior is about the automatic creation of *PTRRecord* objects.

In internal IPWorks:



```
IPWorks> create arecord internalarea test1.example.net 10.0.0.5
Would you like to create a PTR Record for the assignment of the address "10.0.0.5" to the
dnsname "test1.example.net" (it will be added to zone "10.in-addr.arpa")? [yes] >
1 object(s) created.
IPWorks> create arecord internalarea test2.example.net 10.1.0.5
1 object(s) created.

IPWorks> list ptrrecord -format=brief
[PTRRecord 5.0.0.10.in-addr.arpa -> test1.example.net]
```

In external IPWorks:

```
IPWorks> create arecord externalarea test3.example.net 12.0.0.5
1 object(s) created.
IPWorks> list ptrrecord -format=brief
[PTRRecord 5.0.0.12.in-addr.arpa -> test3.example.net]
```

Creating subnet policies are only applied to the *ARecord* objects that are created after the subnet is created. If the *ARecord* objects have already been defined using addresses in these subnets, the policies will not be retroactively applied.

3.1.3.2 Creating GenRecords

The following is an example of a reverse lookup zone for a class C network (1.2.3.0/24) that consists entirely of generated *PTRRecord* objects.

```
IPWorks> create masterzone indns1 3.2.1.in-addr.arpa
1 object(s) created.
IPWorks> create genrecord -prompt
Area? [next] >
Start? [next] > 1
End? [next] > 254
Step? [next] >
DnsNameTemplate? [next] > $.3.2.1.in-addr.arpa
TTL? [next] >
Type? [next] > PTR
RDataTemplate? [next] > host$.example.com
BaseDnsName? [next] >
Description? [next] >
1 object(s) created.
```

Verify the result:

```
IPWorks> list -format=conf
$GENERATE 1-254 $.3.2.1.in-addr.arpa. PTR
host$.example.com
```

3.1.3.3 Implementing Classless in-addr.arpa Delegation

The example that follows is presented to illustrate how to implement this solution using IPWorks. It is not an attempt to explain how the solution works.



Consider a situation where an ISP has split a class C network (1.2.3.0/24) into three smaller networks that are contracted out to three customers, as follows:

- The first customer (example.com) uses the first 64 addresses (1.2.3.0 to 1.2.3.63). The ISP manages DNS for this customer and the customer uses a simple naming scheme where each name of host is based on the address assigned to that host.
- The second customer (example.com) uses the next 64 addresses from to 1.2.3.64 to 1.2.3.127 and they manage their own DNS name space. The name of their DNS Server is ns1.example.com.
- The third customer (example.com) uses the upper 128 addresses (1.2.3.128 to 1.2.3.255) and they also manage their own DNS name space. The name of their DNS Server is ns1.example.com.

First a *dnsserver* object is created and the zone that contains the normal default lookup names:

```
IPWorks> create dnsserver ns1 -set dnsname=ns1.example.com
-set address=192.0.0.1
1 object(s) created.
IPWorks> create masterzone ns1 3.2.1.in-addr.arpa
1 object(s) created.
```

Next the *masterzone* objects for example.com are created. This includes a special subzone of the normal reverse lookup zone that contains the *PTRRecord* objects for example.com.

```
IPWorks> create masterzone ns1 example.com
1 object(s) created.
IPWorks> create masterzone ns1 0-63.3.2.1.in-addr.arpa
1 object(s) created.
```

Both example2.com and example3.com have similar subzones of the normal reverse lookup zone. The two DNS server name will contain the *PTRRecord* objects for those clients. But in this case, the objects are being managed externally. So only creation of the *NSRecord* objects to delegate the subzones is needed.

```
IPWorks> create nsrecord -set
container="ns1:default:3.2.1.in-addr.arpa" -set
dnsname=64-127.3.2.1.in-addr.arpa -set
nameserver=ns1.example2.com
1 object(s) created.
IPWorks>
create nsrecord -set
container="ns1:default:3.2.1.in-addr.arpa" -set
dnsname=128-255.3.2.1.in-addr.arpa -set
nameserver=ns1.example3.com
1 object(s) created.
```



A subnet that defines the *PTRRecord* policy for the address space is created for `example1.com`. This can be done for `example2.com` and `example3.com`, but since no records are created using their portion of the address space this is not necessary.

```
IPWorks> create subnet 1.2.3.0/24 -set
ptrstrategy=delegated -set ptrdomain=0-63.3.2.1.in-addr.arpa
1 object(s) created.
```

The final step is to define *CNAMERecord* for the normal reverse lookup domain names that defines those names as aliases for the new delegated subzones. Rather than enumerating more than 250 *CNAMERecord* objects, the *GenRecord* objects (described in the previous section) are used to accomplish this:

```
IPWorks> create genrecord -set start=1 -set end=62
-set type=CNAME -set dnsnametemplate=$.3.2.1.in-addr.arpa
-set rdatatemplate=$.0-63.3.2.1.in-addr.arpa
1 object(s) created.
IPWorks> create genrecord -set start=65 -set end=126
-set type=CNAME -set dnsnametemplate=$.3.2.1.in-addr.arpa
-set rdatatemplate=$.64-127.3.2.1.in-addr.arpa
1 object(s) created.
IPWorks> create genrecord -set start=129 -set end=254
-set type=CNAME -set dnsnametemplate=$.3.2.1.in-addr.arpa
-set rdatatemplate=$.128-255.3.2.1.in-addr.arpa
1 object(s) created.
```

Finally, one *ARRecord* is created in the namespace of `example1.com` to illustrate how the associated *PTRRecord* (which is automatically created) is configured:

```
IPWorks> create arecord test.example1.com 1.2.3.1
1 object(s) created.
IPWorks> list ptrrecords -format=conf
1.0-63.3.2.1.inaddr.arpa. IN PTR test.example1.com.
```

3.1.3.4 Implementing \$INCLUDE Directive

This section describes how to implement `$INCLUDE` directive based on the following situations:

- masterzone does not exist.

Create masterzone and related includefile and includerecord.

```
IPWorks> create masterzone dns1 iptelco.com
1 object(s) created.
```

```
IPWorks> create includefile -prompt
FileAlias? [next] > 4
LoadForm? [next] > /home/eanycha/include.txt
1 object(s) created.
```

**Note:**

- Make sure that the `include.txt` already exists.
- The field `LoadFrom` in `IncludeFile` must not be configured as “reload”, however `IncludeFile` is allowed to contain the reload as a substring.

```
IPWorks> create includerecord -prompt
Alias? [next] >
BaseDnsName? [next] > dns1.ipstelco.com
FileAlias? [next] > 4
1 object(s) created.
```

Where: 4 represents the name of `includefile`. It can be any number or string. It's recommended to use a new name different from the old one.

- `masterzone` already exists, and the existing `include.txt` has been modified (for example, some records are added in the file).

Modify the `masterzone`, and modify `includefile` to reload include file from the original path.

```
IPWorks>modify MasterZone ipstelco.com -set
ExportNeeded=true
IPWorks>modify includefile 4 -set loadfrom=Reload
```

The keyword `reload` is reserved for reloading include file from the original path.

- `masterzone` already exists, and another include file will be used to replace the original one.

Modify the `masterzone`, and modify `includefile` to connect to another include file.

```
IPWorks>modify MasterZone ipstelco.com -set
ExportNeeded=true
IPWorks>modify includefile 4 -set loadfrom=
<name of includefile>.txt
```

3.1.4 Configuration of Zones

This section describes configuration of zones of types that are not included in any of the sample networks.

The configuration of zones contains the following topics:

- Section 3.1.4.1 Creating a Dynamic Zone on page 24.
- Section 3.1.4.2 Configuring Fixed Zone on page 25.



- Section 3.1.4.3 Configuring ForwardZones on page 26.
- Section 3.1.4.4 Configuring Zones with Hidden Masters on page 26.
- Section 3.1.4.5 Configuring Zones with Multiple Masters on page 28.
- Section 3.1.4.6 Configuring TCP Port 53 Support for DNS+ENUM Deployment on page 29.

3.1.4.1 Creating a Dynamic Zone

Create a dynamic zone in the same way for a static zone, by initially creating a *MasterZone* object inside a server. There is a field of the *MasterZone* objects called *IsDynamic*. It is a boolean flag that indicates whether the zone is a dynamic zone. This can be set when the zone is initially created, or it can be set at any time after that. For it to function properly, it is advised to configure the zone to receive updates from any host by setting the parameter *allow-update* to *{any;}*.

Once this flag is set, the zone is considered to be in a pending state until the server is updated. Since the server has not been updated, the zone is still being managed with the IPWorks database as it is authoritative source. When the server is updated, the dynamic zone is activated. This is the process by which the management of the zone data is handed over from the IPWorks database to the server. The resource records in the zone are copied to the server and they are deleted from the IPWorks database later. When this is complete, the DNS Server has the master copy of the resource records of zone.

The most important aspect of this procedure is to realize that it is effectively a one-way procedure. Although it is easy to convert a static zone to a dynamic zone, it is much more complicated to reverse the procedure. There is no automated method for copying the records back into the IPWorks database. In fact, the only way to restore them is to shut down the server and manually import the records. This can be both tedious and time-consuming so it is important that the zones are not made dynamic unless it needs to be configured in that way.

```
IPWorks> create masterzone -set name=example.com;server=tn0-s3;isdynamic=true
1 object(s) created.
IPWorks> list masterzone example.com
[MasterZone example.com]
  Partition: active
  Server: tn0-s3
  View: default
  Name: example.com
  ZoneId: tn0-s3:default:example.com
  ZoneType: master
  Filename: db.example.com
  DefaultTtl: 86400
  ExportNeeded: true
  SourceName: tn0-s3.alfa.noc
  AuthoritativeName: tn0-s3.alfa.noc
  MasterZoneType: File
  IsDynamic: true
  Option: allow-query { key tn0s3-default-smkey; any; },
  allow-transfer { key tn0s3-default-smkey; any; },
  allow-update { key tn0s3-default-smkey; }
```



Note: Within one area, different views cannot have the same zone if at least one of the zones is a dynamic zone.

3.1.4.2

Configuring Fixed Zone

Each of these objects has a field called `fixed zone`. To add a fixed zone, add the name of the `fixedzone` to this field and that fixed configuration of zone configuration will be included with that object configuration. This makes it easier to share these zones across views and across servers, without having to create many instances of the same zone or resource records or both across many servers.

```
IPWorks> select dnsserver ns0
Selected 1 object(s).
IPWorks> modify -add fixedzone=loopback
Working on 1 object(s).
1 object(s) were updated.
```

Note: If fixed zones are not being used, own *MasterZone* objects for the zones that are commonly implemented as fixed zones can be created, such as the `localhost` or `loopback` zones.

Configuring LocalhostZones

This example shows the standard configuration for the *LocalhostZone* object:

```
IPWorks> select fixedzone localhost
Selected 1 object(s)
IPWorks> list -format=zonefile
; This is the standard definition of the localhost zone
;
@ IN SOA localhost. root.localhost. (
    2002112601 ; Serial
    3600       ; Refresh
    300        ; Retry
    604800     ; Expire
    86400 )    ; Minimum
    IN NS localhost.
localhost.IN A 127.0.0.1 ; Local loopback
localhost.IN AAAA 0000:0000:0000:0000:0000:0000:0000:0001
```

Configuring LoopbackZones

This example shows the standard configuration for the IPv4 *LoopbackZone*:

```
IPWorks> list fixedzone loopback -format=zonefile
; This is the standard definition of the loopback zone
;
@ IN SOA localhost. root.localhost. (
    2002112601 ; Serial
    3600       ; Refresh
    300        ; Retry
```



```
        604800      ; Expire
        86400 )      ; Minimum
    IN NS localhost.
1 IN PTR localhost. ; Local loopback
```

3.1.4.3 Configuring ForwardZones

This is an example of a *ForwardZone* declaration:

```
IPWorks> create forwardzone ns0 example.com -prompt
View? [next] >
Option? [next] > forwarders { 1.2.3.1; 1.2.3.2; 1.2.3.4; }
Option: forwarders { 1.2.3.1; 1.2.3.2; 1.2.3.4; }
Option? [next] >
Description? [next] >
1 object(s) created.
```

3.1.4.4 Configuring Zones with Hidden Masters

With IPWorks, it is possible to manually edit or delete the resource records if working at that level is preferred. However, a simple method can also be adopted to configure a zone to have a hidden master. This is done by editing two fields on the *MasterZone* object itself, as done in the following example:

First three DNS servers are created for the example, then one of the servers is set as the master server for the zone *example.com* and the other two servers are set as slaves for this zone. The following commands set this up:



```

IPWorks> create dnsserver ns1 -set dnsname=ns1.example.com -set address=12.1.0.1
1 object(s) created.
IPWorks> create dnsserver ns2 -set dnsname=ns2.example.com -set address=12.2.0.1
1 object(s) created.
IPWorks> create dnsserver ns3 -set dnsname=ns3.example.com -set address=12.3.0.1
1 object(s) created.
IPWorks> create masterzone ns1 example.com
1 object(s) created.
IPWorks> create slavezone ns2 example.com -set source=12.1.0.1
The zone example.com is a slave to [DnsServer ns1] which
declares the zone in view default. Should this slave zone be
advertised with an NS record as an authoritative source for
the zone? [yes] >
1 object(s) created.
IPWorks> create slavezone ns3 example.com -set source=12.1.0.1
The zone example.com is a slave to [DnsServer ns1] which
declares the zone in view default. Should this slave zone be
advertised with an NS record as an authoritative source for
the zone? [yes] >
1 object(s) created.
IPWorks> select masterzone ns1 example.com
Selected 1 object(s).
IPWorks> list
[MasterZone example.com]
Partition: active
Server: ns1
View: default
Name: example.com
ZoneId: ns1:default:example.com
ZoneType: master
Filename: db.example.com
DefaultTtl: 86400
ExportNeeded: true
SourceName: ns1.example.com
AuthoritativeName: ns1.example.com, ns2.example.com,
                  ns3.example.com
IsDynamic: false

```

At this point, the zone is setup using the *normal* configuration for a zone. Two changes are made to the zone. First the master servers dnsname from the list of authoritative names are deleted. These are the names that are advertised in the *NSRecord* objects for this zone. By removing the name, the corresponding *NSRecord* is automatically deleted. This can be configured when the *MasterZone* is originally created and if the names of the servers are known then.

The servers *source name* is also deleted. This is the dnsname that is advertised in the *SOARecord*. This is not as critical as removing the authoritative name, but it is done to make sure that the master is hidden.

```

IPWorks> modify -remove authoritativeName=ns1.example.com
Working on 1 object(s).
1 object(s) were updated.
IPWorks> modify -set sourceName=ns2.example.com
Working on 1 object(s).
1 object(s) were updated.

```

Now that these changes made can be looked at the zonefile for the zone to confirm that the master is hidden:



```
IPWorks> list -format=zonefile
; Zone file for "example.com" in "default" view
; for server "ns1".
;
; This file was generated by IPWorks at 03/11/06 22:29:28
;
; It should not be edited directly because any changes
; will be overwritten by IPWorks the next time the
; servers configuration is updated in IPWorks.
;
$TTL 86400
example.com.      IN SOA ns2.example.com.
                  root.ns1.example.com. 1 10800 3600 604800 86400
; NS Records for authoritative servers:
example.com.      IN NS ns2.example.com.
example.com.      IN NS ns3.example.com.
; Resource Records:
```

3.1.4.5 Configuring Zones with Multiple Masters

The following example shows a simple zone with multiple masters:

First two DNS servers are created and a *MasterZone* in each server with the same name is created. What makes this work is that each of the *MasterZone* objects is associated with the same area (in this case it is the default area). This means that any time a record is added to the area it will be automatically added to each of the zones. Precaution must be taken to ensure that the serial numbers in the *SOARecord* objects are maintained so they are the same. But this will happen automatically as long as the servers are always updated together.

```
IPWorks> create dnsserver ns1 -set dnsname=ns1.example.com -set address=12.1.0.1
1 object(s) created.
IPWorks> create dnsserver ns2 -set dnsname=ns2.example.com -set address=12.2.0.1
1 object(s) created.
IPWorks> create masterzone ns1 example.com
1 object(s) created.
IPWorks> create masterzone ns2 example.com
1 object(s) created.
```

Now that an *ARecord* is created and placed in the default area. It is included in both zones automatically. To confirm this, the zonefile is listed for each of the zones to see if they are still identical (except for the records that advertise the authoritative server for the zone).



```
IPWorks> create arecord test.example.com 12.3.1.1
1 object(s) created.
IPWorks> select masterzone ns1 example.com
Selected 1 object(s).
IPWorks> list -format=zonefile
; Zone file for "example.com" in "default" view
for server "ns1".
;
; This file was generated by IPWorks at 03/11/06 23:01:25
;
; It should not be edited directly because any changes
; will be overwritten by IPWorks the next time the
; server's configuration is updated in IPWorks.
;
$TTL 86400
example.com.      IN SOA ns1.example.com.
                  root.ns1.domain.com. 1 10800 3600 604800 86400
; NS Records for authoritative servers:
example.com.      IN NS ns1.example.com.
; Resource Records:
test.example.com. IN A 12.3.1.1

IPWorks> select masterzone ns2 example.com
Selected 1 object(s).
IPWorks> list -format=zonefile
; Zone file for "example.com" in "default" view
for server "ns2".
;
; This file was generated by IPWorks at 03/11/06 23:01:44
;
; It should not be edited directly because any changes
; will be overwritten by IPWorks the next time the
; server's configuration is updated in IPWorks.
;
$TTL 86400
example.com.      IN SOA ns2.example.com.
                  root.ns2.domain.com. 1 10800 3600 604800 86400
; NS Records for authoritative servers:
example.com.      IN NS ns2.example.com.
; Resource Records:
test.example.com. IN A 12.3.1.1
```

3.1.4.6 Configuring TCP Port 53 Support for DNS+ENUM Deployment

To configure TCP port 53 support for DNS+ENUM deployment if required:

1. Check whether listen-on port 53 {any;} tcp-only yes is configured on SS.

```
# ipwcli
```

```
IPWorks> list dnsserver dns1
```

2. Modify the dnsserver object.

```
IPWorks> modify dnsserver dns1 -add option="listen-on
port 5300 {any;}", "listen-on port 53 {any;} tcp-only
yes"
```

3. Update the DNS Server.

```
IPWorks> update dnsserver dns1
```



3.1.5 Configuring DNS Caching

DNS Caching resolves Internet domain queries from the end-user devices. When a query is received, a DNS caching server firstly examines whether the queried domain name is already in the cache.

The examination procedure is as following:

- If yes (cache hit), the cached answer is replied directly.
- If not (cache miss), the caching DNS starts query from root hints until an authoritative name server for the queried domain is found, or forwards the query to forwarders.

IPWorks DNS server can be configured to use root hints, forwarders, or both to provide DNS Caching.

- Section 3.1.5.2 Configuring IPWorks DNS to Use Root Hints on page 31
- Section 3.1.5.3 Configuring IPWorks DNS to Use Forwarders on page 31;
- Section 3.1.5.4 Configuring IPWorks DNS to Use both Root Hints and Forwarders on page 31.

After finishing the configuration, ensure to disable the debug logging in order to avoid performance impact of logging. Refer to Section 3.4.6 on page 48 *Viewing Server Logs* to check and disable the DNS server logs.

3.1.5.1 IPWorks DNS Common Configuration

1. Create a DNS server.

For example, the OAM IP of IPWorks DNS server is 10.175.20.20.

```
IPWorks> create dnsserver dns1 -set address=10.175.20.20
```

2. Create an ACL.

For example, the allowed client subnets (used for DNS ACL configuration) are 172.128.0.0/21, 172.100.0.0/18.

```
IPWorks> create acl GIACL -set matchlist="{172.128.0.0/21; 172.100.0.0/18;}"
```

3. Add related options list for the DNS server.

For example, the traffic IP of IPWorks DNS server is 172.24.10.10.

```
IPWorks> modify dnsserver dns1 -add option="allow  
-query-cache {127.0.0.1;GIACL;}", "allow-recursion  
{127.0.0.1;GIACL;}", "allow-transfer {none;}", "listen-on  
{127.0.0.1; 172.24.10.10;}", "query-source address
```



```
172.24.10.10", "recursion true", "recursive-clients
10000", "minimal-responses yes"
```

3.1.5.2 Configuring IPWorks DNS to Use Root Hints

1. IPWorks DNS common configuration, see Section 3.1.5.1 IPWorks DNS Common Configuration on page 30.
2. Modify `hintzone internet`.

IPWorks has a default `hintzone` named `internet` which includes root hints. You can update `hintzone` content if the root hints are not latest. For the latest Internet root hints, refer to <http://www.internic.net/domain/named.root>.

```
IPWorks> modify hintzone internet -set contents="\n.
3600 IN NS <root FQDN 1>.\n. 3600 IN NS <root FQDN
2>.\n<root FQDN 1>. 6D IN A xx.xx.xx.xx\n<root FQDN
2>. 6D IN A yy.yy.yy.yy"
```

3. Modify the `fixedzone` used by the DNS server.

```
IPWorks> modify dnsserver dns1 -add fixedzone=internet
```

4. Update the DNS server to make the configuration take effect.

```
IPWorks> update dnsserver dns1
```

3.1.5.3 Configuring IPWorks DNS to Use Forwarders

1. IPWorks DNS common configuration, refer to Section 3.1.5.1 IPWorks DNS Common Configuration on page 30.
2. Add option `forwarders`.

DNS server forwards the query to the other public Internet DNS server according to configured sequence.

```
IPWorks> modify dnsserver dns1 -add option="forwarders
{< public DNS Server address1>;<public DNS Server
address2>;<...>;}"
```

3. Add option `forward only`.

```
IPWorks> modify dnsserver dns1 -add option="forward
only"
```

4. Update the DNS server to make the configuration take effect.

```
IPWorks> update dnsserver dns1
```

3.1.5.4 Configuring IPWorks DNS to Use both Root Hints and Forwarders

1. Configure root hints, see Section 3.1.5.2 Configuring IPWorks DNS to Use Root Hints on page 31.



2. Configure forwarders, see Section 3.1.5.3 Configuring IPWorks DNS to Use Forwarders on page 31.

Use option `forward first` instead of `forward only`.

```
IPWorks> modify dnserver dns1 -add option="forward
first"
```

3.1.5.5 Configuring Internet DNS

Note:

- This feature is only valid for IPWorks Virtual Deployment for KVM.
- Before configuration, make sure the license of Internet DNS Feature is ready. For details, refer to *License Management*.
- Before configuration, make sure the DNS caching is configured. For more detail, refer to Section 3.1.5 Configuring DNS Caching on page 30.

You can do the following to configure the Internet DNS function:

- To enable the Internet DNS function, do the following:

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1
,IpworksDnsRoot=1,DnsServer=1,BindService=1

(config-BindService=1)>internetDNS=true
```

- To disable the Internet DNS function, do the following:

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1
,IpworksDnsRoot=1,DnsServer=1,BindService=1

(config-BindService=1)>internetDNS=false
```

Note: Restart the DNS server to make the changes take effect.

3.2 Configuring ASDNS

This section guides the configuration personnel how to configure ActiveSelect DNS.

Prerequisites:

Before configuring ASDNS, ensure that the Storage Server is started. If the Storage Server is not started, execute the following commands:

1. Log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>
```



2. Start Storage Server on both SC-1 and SC-2.

```
#ipw-ctr start ss <SC hostname>
start ss ==> success.
```

Actions:

The configuration of ASDNS contains the following topics:

- Creating ASDNSSite, see Section 3.2.1 on page 33
- Configuring ASDNSPolicy, see Section 3.2.2 on page 33
- Creating ASDNSRecord, see Section 3.2.3 on page 34
- Creating Monitor, see Section 3.2.4 on page 35
- Displaying MonitorScript, see Section 3.2.5 on page 35
- Configuring MonitorMethod, see Section 3.2.6 on page 36
- Configuring DnsContact, see Section 3.2.7 on page 38
- Creating MonitorResource, see Section 3.2.8 on page 39

All actions from the list are based on the *Sample Network 1* configurations in the Section 3.1 on page 5.

3.2.1 Creating ASDNSSite

In *Sample Network 1* described in Section 3.1.1 Configuring DNS in Sample Network 1 on page 5, the two sites are defined - one to represent the East LAN and one to represent the West LAN. They are both defined in the `internal` area, see the following commands for example:

```
IPWorks> create asdnssite internalarea EastLAN
-set address=10.1.0.0/16
1 object(s) created.
IPWorks> create asdnssite internalarea WestLAN -set address=10.2.0.0/16
1 object(s) created.
IPWorks> list asdnssite
[ASDNSSite westlan]
  Partition: active
  Area: internalarea
  Name: WestLAN
  Address: 10.2.0.0/16
```

Note: The output of the `list` command is truncated to display only one *ASDNSSite*. However, it displays all the relevant *ASDNSSite* objects (in this case, it is 2: EastLAN and WestLAN).



3.2.2 Configuring ASDNSPolicy

Before the user creates an *ASDNSPolicy* object, any *ASDNSSite* objects that will be referenced by that policy must be created.

In *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), policy that will be used by all the *ASDNSRecord* objects is defined. The following rules are followed:

1. Queries from the East LAN (10.1.0.0/16) try to use APNs from the East LAN if they are available. Otherwise, fall back to using the APNs from the West LAN.
2. Queries from the West LAN (10.2.0.0/16) try to use APNs from the West LAN if they are available. Otherwise, fall back to using the APNs from the East LAN.
3. All other queries use the default behavior for resolving names.

Take the following commands as an example for defining the *ASDNSPolicy* for the rules above:

```
IPWorks> create asdnspolicy internalarea Pref
erLocalApns -set
prefer="10.1.0.0/16 prefer EastLAN WestLAN",
"10.2.0.0/16 prefer WestLAN EastLAN"
1 object(s) created.
IPWorks> list
[ASDNSPolicy preferlocalapns]
Partition: active
Area: internalarea
Name: PreferLocalApns
Prefer: 10.1.0.0/16 prefer "EastLAN" "WestLAN",
        10.2.0.0/16 prefer "WestLAN" "EastLAN"
```

3.2.3 Creating ASDNSRecord

In the *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), there are only two APNs for which ActiveSelect is enabled. Since they have the same address bindings, one of them is defined as an alternative dnsname for the other. This can be done by creating the following *ASDNSRecord*:

```
IPWorks> create asdnsrecord internalarea Inter
net.mnc001.mcc001.gprs
-set policy=PreferLocalApns
-set alternatednsname=www.mnc001.mcc001.gprs
1 object(s) created.
```



To view the ASDNS rule that implements the policy, it can be displayed as follows:

```
IPWorks> list -format=conf
resource "Internet.mnc001.mcc001.gprs" A {
  dnsname "Internet.mnc001.mcc001.gprs";
  alt_dnsname "www.mnc001.mcc001.gprs";
  site "EastLAN" {
    10.1.0.0/16;
  };
  site "WestLAN" {
    10.2.0.0/16;
  };
  10.1.0.0/16 prefer "EastLAN" "WestLAN";
  10.2.0.0/16 prefer "WestLAN" "EastLAN";
};
```

3.2.4 Creating Monitor

In the *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), only one *monitor* is running. To keep things simple, the *monitor* will be run on the same system as the internal master DNS Server.

```
IPWorks> create monitor indns1mon -set dnsname=indns1
.example.net -set address=169.254.101.3
1 object(s) created.
IPWorks>list
[Monitor indns1mon]
  Partition: active
  Name: indns1
  Address: 169.254.101.3
  DnsName: indns1.example.net
  Filename: asdnsmon.conf
  Type: Monitor
  ExportNeeded: true
```

Note: The output does not include the status field as Status: On <date> at <time> server is <down/up>. When the *monitor* is created but had not as yet started the Server Manager even once for the Server Manager. Each time we issue a list it will show the status as up or down depending on the server status.

If IPWorks *monitor* is installed on a system in the network, it can be configured to *auto-register* itself when the Server Manager for that *monitor* first starts up. It will create a *Monitor* object in the database to correspond to the active *monitor*. It will be defined with the IP addresses and domain names for that system. This feature allows creating these objects manually. The automatically created objects need to be reviewed for the *Monitor*.



3.2.5 Displaying MonitorScript

In the *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), the PING script is used to monitor the status of the GGSN node. The following commands display information about the *MonitorScript* objects:

```
IPWorks> list monitorscript -format=brief
[MonitorScript pingmonitor]
[MonitorScript epdgmonitor]
[MonitorScript snmpmonitor]
IPWorks> list monitorscript pingmonitor -format=conf
#!/bin/sh
# -----
#
# COPYRIGHT Telefonaktiebolaget L M Ericsson 2013
#
# The copyright of the computer program herein is the
# property of Telefonaktiebolaget L M Ericsson, Sweden.
# The program may be used and/or copied only with the
# written permission from Telefonaktiebolaget L M Ericsson
# or in the accordance with the terms and conditions
# stipulated in the agreement/contract under which the
# program has been supplied.
#
# -----
#
# Module Description:
# Script to monitor an IP Address with ping;
# return exit status of ping
# for the IPworks ASDNS Monitor
#
USAGE="usage: pingmonitor <address> [timeout]"
if [ $1 ]
then
    ping $1 $2
else
    echo $USAGE
fi

exit $?
```

ASDNS MonitorScripts is placed in the directory `/etc/ipworks/<PL hostname>/asdnsmon`.



3.2.6 Configuring MonitorMethod

3.2.6.1 Creating MonitorMethod with the PingMonitor Script

In the *Sample Network 1* described in Section 3.1.1 Configuring DNS in Sample Network 1 on page 5, the predefined `PingMonitor` script is used and the status of the GGSN is checked every 5 minutes by creating the `MonitorMethod` object.

```
IPWorks> create monitormethod pingevery5minutes -set interval=5m -set filename=pingmonitor -set argument=@address
1 object(s) created.
IPWorks> list
[MonitorMethod pingevery5minutes]
  Partition: active
  Name: pingevery5minutes
  Type: status
  Filename: pingmonitor
  Interval: 300
```

3.2.6.2 Creating MonitorMethod with the EpdgMonitor Script

In this section, an example is provided to create `MonitorMethod` with `epdgmonitor` script. In `epdgmonitor`, there are three parameters need to be set into field `argument`.

In this example, The first parameter is set as `@address`. The ePDG Route Processor (RP) address `10.0.0.1` is set as the second parameter. The SNMP community string `public` is set as the third parameter.

```
IPWorks> create monitormethod epdgmethod -set interval=30s -set filename=epdgmonitor -set argument=@address,10.0.0.1,public
```

```
1 object(s) created.
```

```
IPWorks> list monitormethod epdgmethod
```

```
[MonitorMethod epdgmethod]
  Partition: active
  Name: epdgmethod
  Type: status
  Filename: epdgmonitor
  Interval: 30
  Argument: @address, 10.0.0.1, public
```

Note: Adding the route for SNMP messages between IPWorks Traffic IP and ePDG OAM IP is needed for this function. It might take potential risk to the network security. So, some policy rules in firewall is needed to keep the network security.



3.2.6.3 Creating MonitorMethod with the SnmpMonitor Script

Following the examples are provided to create `MonitorMethod` with `snmpmonitor` script. The script can be used to get the status and load of an monitored resource with a specified IP address.

- Status monitoring

```
IPWorks>create monitormethod snmpmethod -set
interval=30s -set filename=snmpmonitor -set
argument=@address
```

```
1 object(s) created.
```

```
IPWorks> list monitormethod snmpmethod
```

```
[MonitorMethod snmpmethod]
Partition: active
Name: snmpmethod
Type: status
Filename: snmpmonitor
Interval: 30
Argument: @address
```

- Load monitoring

```
IPWorks>create monitormethod snmpmethod -set
interval=30s -set filename=snmpmonitor -set
argument=@address -set type=load
```

```
1 object(s) created.
```

```
IPWorks>list monitormethod snmpmethod
```

```
[MonitorMethod snmpmethod]
Partition: active
Name: snmpmethod
Type: load
Filename: snmpmonitor
Interval: 30
Argument: @address
```

3.2.7 Configuring DnsContact

IPWorks automatically creates a `DnsContact` object when a new DNS Server is created. This behavior makes a better interaction with the DNS server and `DnsContact`. The name for the contact is the same as the name for the server and the contact is defined to use the primary address of the server. If no address is specified, a `DnsContact` is not created.



In the *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), the *DnsContact* automatically created for the master internal DNS Server (indns1) is used. See the following commands for the definition of the default contact:

```
IPWorks> list dnscontact indns1
[DnsContact indns1]
  Partition: active
  Name: indns1
  Server: indns1
  Address: 169.254.101.3
  MonitorDeclarer: indns1mon
```

The default value of TCP/IP port in *DnsContact* is port 53. But if the DNS server is running with the ENUM server on the same machine, the DNS server will use port 5300 instead of port 53. Therefore, the *DnsContact* value should be specified to port 5300.

```
IPWorks> select dnscontact
Selected 1 object(s).
IPWorks> modify -add port=5300
Working on 1 object(s).
1 object(s) were updated.
IPWorks> update dnsserver
Result of performing an export is:
  Skipped the zone [MasterZone test.com] - no export needed
  Exported ASDNS configuration for [DnsServer dns1]
  Exported configuration for [DnsServer dns1]
  Updated the configuration for 'DNS' server 'dns1'.
IPWorks> update monitor
Result of performing an export is:
  Exported [MonitorScript pingmonitor]
  Exported configuration for [Monitor monitor1]Updated the configuration for 'ASDNSMON' server 'monitor1'.
```

3.2.8 Creating MonitorResource

In the *Sample Network 1* (see Section 3.1.1 Configuring DNS in Sample Network 1 on page 5), there are two *resources* (the GGSN nodes) that must be monitored by creating two *MonitorResource* objects, as follows:

```
IPWorks> create monitorresource indns1mon ggsn1 -set address=10.1.0.2
-set method=pingevery5minutes -set reportcontact=indns1
1 object(s) created.
IPWorks> create monitorresource indns1mon ggsn2 -set address=10.2.0.2
-set method=pingevery5minutes -set reportcontact=indns1
1 object(s) created.
IPWorks> list
[MonitorResource ggsn2]
  Partition: active
  Monitor: indns1
  Name: ggsn2
  Address: 10.2.0.2
  Method: pingevery5minutes
  ReportContact: indns1
```



3.3 DNS Server IPv6 Support

The IPv6 support of IPWorks DNS server covers the following aspects:

- IPWorks DNS Server supports the IPv4/IPv6 Dual Stack on the traffic interface, which makes it possible to use IPWorks DNS query/responses on IPv6 transport plane.
- IPWorks DNS Server supports the AAAA Resource Records on the traffic plane, and IPv6 reverses lookups by PTR Resource Records, DNS Zone Transfer, and View/ACL control functions over IPv6.
- IPWorks DNS server supports queries on both the IPv4 and IPv6 transports as specified by listen-on option and listen-on-v6 option.

Note: For detailed descriptions on `listen-on` option and `listen-on-v6` option, refer to the Section *DNS Options* in *IPWorks Configuration Management*.

When the option is for the DNS server to listen the incoming queries using the different Internet Protocols, configure the option as follows:

- Queries using IPv4: `listen-on`
- Queries using IPv6: `listen-on-v6`
- Queries using both IPv4 and IPv6: both `listen-on` and `listen-on-v6`

```
IPWorks> create dnsserver <name> -set
address=169.254.100.3 or 169.254.100.4;
option="listen-on {any;}", "listen-on-v6 {any;}"
```

- When creating a *DnsServer* object, it is configured with IPv4 support by default. If IPv6 support is needed:
 - Do not use the static IPv6 address, but use the static IPv4 address and a IPv6 address which was added in the additional addresses.
 - DNS Parameters need no configuration.
 - AAAA records need no configuration, but the DNS Server must add the `listen-on-v6` option in `ipwcli` and update it:

```
modify dnsserver dns1 -add
option="listen-on-v6 {any;}"
```

- The interface needs no other configuration.

Table 2 shows examples of configuring DNS IPv6 Support via IPWCLI commands.



Table 2 DNS IPv6 Support Configuration

| Scenario | DNS Port | IPWCLI Command |
|-----------------|----------|---|
| DNS behind ENUM | 5300 | <ul style="list-style-type: none"> If <i>DnsServer</i> is not created: <code>create dnsserver <name> -set address=<169.254.100.3/169.254.100.4>; option="listen-on 5300{any;}"</code> If <i>DnsServer</i> is created: <code>modify dnsserver <name> -set option="listen-on port 5300{any;}"</code> |
| DNS Standalone | 53 | <ul style="list-style-type: none"> If <i>DnsServer</i> is not created: <code>create dnsserver <name> -set address=<169.254.100.3/169.254.100.4>; option="listen-on 53{any;}", "listen-on-v6 53{any;}"</code> If <i>DnsServer</i> is created: <code>modify dnsserver <name> -set option="listen-on port 5300{any;}"</code> |

To support IPv6 connection between DNS client and DNS server, the following command needs to be added in *Evip* by using ECLI.

```
command="ip -6 route add <DNS Client IPv6 subnet>/<prefix length> dev ipw_sig_sp src <VIP_TRF_IP IPv6 address>"
```

For details about how to add a *Evip* command in ECLI, refer to section *Add Route for IPWorks PL Node* in *Configure Route for IPWorks PL Node*.

3.4 DNS Operation

This section describes common operation instructions for the DNS server. The following DNS-related operations can be performed by the user after the DNS server is configured according to the instruction above:

- Starting and Stopping DNS Server, see Section 3.4.1 on page 41
- Dynamic DNS Update using *nsupdate*, Section 3.4.2 on page 42
- Managing Records in Dynamic Zone, see Section 3.4.3 on page 44
- Updating Servers, see Section 3.4.4 on page 46
- Showing Information on Configuration Options, see Section 3.4.5 on page 47
- Viewing Server Logs, see Section 3.4.6 on page 48
- Importing and Exporting Data, see Section 3.4.7 on page 49



3.4.1 Starting and Stopping DNS Server

IPWorks provides mechanisms for controlling the DNS server once it is installed and operating. Users can use the `ipw-ctr` command to start or stop the server directly from the system where the server is in operation.

The following example shows how to stop and start the DNS server:

1. Log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>  
Password:<Password>
```

2. Start the DNS server.

```
#ipw-ctr start dns <PL hostname>  
Start dns ==> success.
```

3. Stop the DNS server.

```
#ipw-ctr stop dns <PL hostname>  
Stop dns ==> success.
```

Attention!

In DNS/ENUM co-location environment, to avoid DNS traffic loss, the operator shall stop ENUM server, and then stop DNS server (in the same PL with the ENUM server).

In DNS/ENUM co-location environment, paired DNS/ENUM servers are running on both PLs. When DNS server is stopped and ENUM server is still running on the same PL at the same time, DNS server loses half traffic. In the PL, ENUM listens on port 53 and DNS listens on port 5300, ENUM forwards DNS requests to DNS through port 5300 internally. So, from eVIP point of view, the port 53 is always alive. eVIP keeps forwarding DNS packages to ENUM server even though the co-located DNS server is stopped, then it will cause DNS traffic loss.

3.4.2 Dynamic DNS Update Using nsupdate

Dynamic DNS updates using the `nsupdate` utility proceeds as follows:

Note: Dynamic DNS update is not fully supported. The update will not spread to all the blades.

- Users issue `nsupdate` command from CLI to add or delete records. A file with a set of commands can also be used for this purpose. For more



information on CLI operations, refer to *Command File Support* in *Command Line Interface User Guide for IPWorks SS*.

- Only `update add` and `update delete` commands are supported: thus, an update is effected by deleting a record (`update delete`) and then update-adding a new record with modified information (for example, `update add arecord`).
- The changes are stored in the journal file of the zone. A journal file is created when the first update takes place.
- The contents of the journal file are periodically dumped into the zone file. The interval (or schedule) for updating journal entries into the zone file is specific to each DNS Server. The interval for the zone file update depends on the DNS implementation. However, the information available with the DNS is always up-to-date, since the hash table includes entries that are dynamically updated.
- When a server is restarted after a crash, it replays the journal file to incorporate any changes that occurred in the journal file since the last dump.
- The dynamic update is authenticated through the CLI, so the user must have appropriate permission to write data to the DNS Server. It does not use Transaction Signatures (TSIG).
- This mechanism makes updates to the DNS Server much faster since it does not involve the Storage Server.
- The updates are performed on the Primary DNS Server only.
- In a typical `nsupdate` implementation, if the server is not specified, it is inferred from the zone. If the zone is not specified, it is inferred from the other details. The domain name system server is resolved based on the `resolv.conf` file.

To delete the dependency on the `resolv.conf` file in IPWorks, which is not appropriate sometimes, specifying the server name is made mandatory. The server name is the first command or the first line in the file specified along with the `nsupdate` command.

The following is an example to update DNS dynamic:

1. Check whether the Dynamic DNS Update is enabled. If not, enable it.
 - a Check whether the Dynamic DNS Update is enabled.



```
IPWorks> list masterzone
Partition: active
Server: dns1
View: default
Name: iptelco.com
ZoneId: dns1:default:iptelco.com
ZoneType: master
Filename: db.iptelco.com
DefaultTtl: 86400
ExportNeeded: false
SourceName: dns1.iptelco.com
AuthoritativeName: dns1.iptelco.com
MasterZoneType: File
IsDynamic: false
```

The example shows that the Dynamic DNS Update is disabled.

- b Enable the Dynamic DNS Update.

```
IPWorks> modify masterzone iptelco.com -set
isdynamic=true;option="allow-update {any;}"
```

- c Update the changes.

```
IPWorks> update dnsserver dns1
```

- 2. Update an ARecord.

```
IPWorks> nsupdate
>server 127.0.0.1
>update delete host.iptelco.com A
>update add host.iptelco.com 86400 A 10.0.0.3
>send
UPDATE successfull , No. of records updated=2
```

3.4.3 Managing Records in Dynamic Zone

3.4.3.1 Searching for Records in Dynamic Zone

One of the minor differences in working with dynamic zones and static zones is in the way the records are retrieved from a dynamic zone. For most actions in the IPWorks CLI, when a search is performed, that search is limited to the records that are located in the central IPWorks database. Searching through the records in a dynamic zone can be time-consuming (depending on the search criteria) but can also have an impact on the servers performance. For that reason, the dynamic zones are only included in searches when it is needed.



To accomplish this, the searches allow the specification of a *source* as an additional argument to use when searching. To specify this, the zone id should be specified for the dynamic zone. The zone id is a set of three values that uniquely identifies each zone managed by IPWorks. It consists of the server name, the view name, and the zone name, all appended together with a `:` character between each of the consecutive values. An example of a zone id is `ns0:example.com`. If the zone name is unique the name can be just specified as the source for the search, otherwise the entire zone id should be specified.

In the following example, the first search does not match any records because no source is specified so it defaults to searching the central IPWorks database. However, the second search is successful because a source is specified.

```
IPWorks> select arecords -where zone=example.com
No matching object(s) found.
IPWorks> select arecords -source=example.com
Selected 12 object(s).
```

There is one type of search where there is no need to specify a source — if the user is searching for resource records in a dynamic zone using a relationship, then IPWorks can infer that since the zone is dynamic the resource records must use that dynamic zone as the source for the search.

In this example, a relationship that implies the source of the search is used:

```
IPWorks> select masterzone example.com
Selected 1 object(s).

IPWorks> select masterzone example.com -related
resourcerecord -where type="A"
Selected 12 object(s).
```

3.4.3.2 Editing Resource Records in Dynamic Zone

With IPWorks, changing the zone data in a dynamic zone is (for the most part) the same as changing the zone data for static zones. Editing resource records (adding, deleting and modifying them) is performed the same way. If resource records are being edited or deleted, the *source* must be specified in the same way that it was specified for dynamic zone searches. Once the record is located, any modifications performed are done through DDNS updates to the server.

If a new resource record is being created in a dynamic zone, IPWorks will determine that the record is part of a dynamic zone based on the `dnsname` in the resource record. This record will then be transferred and added to the zone through a DDNS update sent to the server.

In both of these cases the DDNS updates are coordinated through the *Server Manager* for that DNS Server. This means that in order to make changes to the data in a dynamic zone both the DNS Server and its Server Manager must



be available. If they are not available, there is a warning when attempting these operations.

3.4.4 Updating Servers

Whenever a server is configured using the IPWorks CLI, the changes are not made in the network until they are activated by updating the server. This allows the changes to be undone, or to prepare a set of changes to be made at a time, before they are activated. However, it is important to ensure that the server must be updated once the changes are made.

Update can be performed in two ways:

1. Update manually when there is a change in the server.
2. Automatically update the server for every fixed period of time.

3.4.4.1 Updating Servers Manually

In this example, a single DNS Server is updated. The `update` procedure examines the data in each zone managed by the server to determine if the zone needs to be updated. If it does not, then it will be skipped.

```
IPWorks> select dnsserver buckeye
Selected 1 object(s).
IPWorks> update
Result of performing an export is:
Exported the zone [LocalhostZone localhost]
Exported the zone [LoopbackZone loopback]
Skipped the zone [MasterZone .] - no export needed
Skipped the zone [MasterZone 117.147.in-addr.arpa] - no export needed
Skipped the zone [MasterZone 5.10.in-addr.arpa] - no export needed
Skipped the zone [MasterZone 5.11.in-addr.arpa] - no export needed
Skipped the zone [MasterZone d3u.us.am.ericsson.se] - no export needed
Skipped the zone [MasterZone example.com] - no export needed
Skipped the zone [MasterZone example.com] - no export needed
Exported ASDNS configuration for [DnsServer buckeye]
Exported configuration for [DnsServer buckeye]
Updated the configuration for 'DNS' server 'buckeye'.
IPWorks>
```

One additional feature of IPWorks is that it is unnecessary to know all the servers that are affected when making a configuration change. However, it is still important to make sure that these servers get updated when the changes are done, otherwise they will not be correctly activated in the network. To resolve this problem, each server has a field on it called **ExportNeeded** and this is automatically set to true by IPWorks when any aspect of that servers configuration is changed.

The **ExportNeeded** field can be used to update all the servers that have pending changes with the command used in the following example. In this example, only one server needed to be updated.

Note: When the command is repeated, the second time it does not update any servers because the flag has been reset and nothing else has changed.



```
IPWorks> update dnsserver -where exportneeded=true
Result of performing an export is:
Exported the zone [MasterZone .]
Skipped the zone [MasterZone 4.10.in-addr.arpa] - no export needed
Skipped the zone [MasterZone example.com] - no export needed
Exported configuration for [DnsServer superkev]
Updated the configuration for 'DNS' server 'superkev'.
IPWorks> update dnsserver -where exportneeded=true
No matching object(s) found.
```

The `rebuild` parameter is only used by the DNS Server to determine whether to rewrite the zone file during the update. If `rebuild` is set to `true`, the zone files are rewritten. If `rebuild` is set to `false`, which is also the default value, only the changed records are appended to the zone files.

```
IPWorks> select dnsserver tn0-s2
Selected 1 object(s).
IPWorks> update dnsserver -rebuild=true
Result of performing an export is:
Exported the zone [MasterZone example.com]
Exported the zone [MasterZone e164.arpa]
Exported the zone [MasterZone example.com]
Exported configuration for [DnsServer tn0-s2]
Updated the configuration for 'DNS' server 'tn0-s2'.
IPWorks>
```

Note:

- Before adding a slave DNS configuration, ensure that the master DNS Server configuration is updated first. If both the master and slave DNS servers are updated simultaneously, the slave server may not receive a complete zone transfer.
- If the operator tries to incrementally update the configuration and provisioning data, use the command `update dnsserver` (without the parameter `-rebuild=true`) instead to save time.

3.4.5 Showing Information on Configuration Options

IPWorks CLI provides a command to display information about a configuration option as shown below:



```

IPWorks> show dns option allow-query
allow-query
Description: Specifies which hosts are allowed to ask
ordinary questions. You may specify allow-query
in the zone statement, in which case it overrides
the options allow-query statement.

Datatype: AddrList
Description: Specifies a list of clients. The syntax
allows you to:
- explicitly specify address(es)
- specify subnet(s) to match
- specify TSIG key(s) to match
- refer to a named ACL
- exclude clients using any of the above criteria.
The syntax is:
<addrlist> = "{" <element> ";"
[<element> ";" ]...}"
<element> = <addrlist> |
<ipaddr> |
<ipnet> |
<aclname> |
"key" <keyid> |
"! " <element>\
<ipaddr> = an IP address
<ipnet> = an IP network in "/masklength" notation
<keyid> = the (quoted) name of a TSIG key
<aclname> = the name of an ACL or one of the
special acl's: (any, none, localhost, localnets)
Example(s): allow-query { 10.0.0.0/8; 192.168/16; 1.2.3.4; }
allow-query { any; }
allow-query { key "key1"; !acl2; localhost; }
Server Config: allow-query <value>;

```

Note: When the user tries to define a value of data type “AddrList” as some ipnet values, IPWorks supports an abbreviative notation so that the zero at the end of the subnet address can be abbreviated. For example, subnet 192.168.0.0/16 could be written as 192.168/16.

If it is unsure on how to define a configuration option for an object, use this command to review the option before setting it. IPWorks check the syntax of any values that has been set for configuration options.

3.4.6 Viewing Server Logs

DNS server provides logging function in operation. By default logging is disabled, operator has to consider the capacity impact when enable logging. Detail operations are as followings:

1. Enable DNS server log using ECLI by configuring the `level` and `debugLogLevel`.

The log level should be `DNS_LOG_LEVEL_DEBUG` and the `debugLogLevel` must be 1~99. For more information about the debug logging level, refer to *debugLogLevel* in *Managed Object Model (MOM)*.



```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,DnsServer=1,BindService=1
>configure
(config-BindService=1)>debugLogLevel=2
(config-BindService=1)>DnsLog=1
(config-log)>level=DNS_LOG_LEVEL_DEBUG
(config-log)>commit
```

2. Obtain the log directory by using ECLI:

```
>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,DnsServer=1,BindService=1
(BindService=1)>show -v
...
log
  directory="/cluster/storage/no-backup/ipworks/logs" <default>
  ...
transactionLog
  directory="/cluster/storage/no-backup/ipworks/logs" <default>
  ...
```

In this example, the logs of the main directories of DNS and DNS transaction are in /cluster/storage/no-backup/ipworks/logs.

3. Go to the directory (/cluster/storage/no-backup/ipworks/logs/<hostname>/) and show the DNS logs and DNS transaction logs:

For example, show the logs on the PL-3:

- `tail -f /cluster/storage/no-backup/ipworks/logs/PL-3/ipworks_dns.log`
- `tail -f /cluster/storage/no-backup/ipworks/logs/PL-3/ipworks_dns_trans.log`

3.4.7 Importing and Exporting Data

This section describes the importing and exporting support that is provided by IPWorks.

Importing refers to the movement of data into IPWorks from an external file, where the data is represented in the file in a known format. Exporting refers to the movement of data from IPWorks into an external file (with a known format), so that the data can be used for a function that is not provided by IPWorks.

Most importing of data is performed when the IPWorks installation is first initialized. A working DNS in place may already be present. Once that data is moved into IPWorks, it will not be necessary to import that data again. For that reason, some of the import functions only works when loading new data. It cannot be used to overwrite, or modify existing data, but only to create new configuration data.



3.4.7.1 Importing DNS Configuration Files

The IPWorks CLI can import DNS configuration files from a BIND 8 or BIND 9 server. This is implemented in the CLI with the `import dns` command. This command can only be used to import a DNS configuration into a new DNS Server object. The object can be created prior to the command, or the import procedure can be used to create the DNS Server object. The latter method is the preferred method because the import procedure can use information in the DNS configuration files to make sure the DNS Server is created correctly.

If the user is importing a configuration into a DNS Server that has been previously created, it must not have any existing configuration information. If it does, this can produce errors during the import. If the definition of the DNS Server is inconsistent with the information in the configuration files, there may be errors. For example, the *NSRecord* and *SOARecord* for the imported zones should use DNS names for the DNS Server object. If they do not, then the import procedure will produce errors. It is necessary to modify the files prior to importing them, to ensure that the imported data is consistent with the server definition.

The behavior of the `import dns` command can be customized by a number of qualifiers. The most commonly used qualifiers are defined here. For further details, (or other qualifiers), consult the online help in the CLI.

The qualifiers are listed below:

- **-server:** Specifies the name of the DNS Server whose configuration is being imported. If it does not already exist, this server will be created.
- **-confonly:** Specifies that only the main configuration file (`named.conf`) should be imported. The zonefiles mentioned in the file will not be processed.
- **-parseonly:** If present, the configuration file(s) will be parsed and checked, but no data will be imported. This can be used for verifying that the files import correctly before performing the operation.
- **-bind8:** If Present, the configuration files are assumed to be BIND 8 configuration files. If not specified, then the importer assumes that they are BIND 9 files. Although the syntax for these two type of files are very similar, there are differences between them. So, it is important to make sure the importer is trying to import the right type of file.

When the import operation is performed, the importer prints informational messages to update on the status of the operation. Additional information can be requested by executing the command using the `-verbose` qualifier (although this can produce very large amounts of output).

Normally the DNS importer starts by reading the `named.conf` file and creating the appropriate configuration objects as defined in that file. It then processes the zonefiles for each of the master zone declarations in the file, importing all



the resource records that are defined in the zonefiles. When the importer is ready to import the zonefiles, it attempts to locate them as follows:

1. If the zonefiles are defined to have a specific location in the `named.conf` file, that location is checked first.
2. If the zonefiles are defined with relative pathnames, the importer uses that relative path based on the location of the `named.conf` file to attempt to locate the zonefile.
3. If the options above do not locate the zonefile, the directory that contains the `named.conf` file is checked to see if it also contains a zonefile of the appropriate name.

The algorithm above is applied to each zone to be imported, one at a time. The first zonefile that is located with this algorithm is the one that is imported.

Example 1 Import DNS Configuration to a new DNS Server

1. Import the `named.conf` file directly to a new DNS server called `dns1`:

```
IPWorks> import dns /var/named/named.conf -server=dns1
Parsing named configuration
Parsing zone 10.in-addr.arpa (db.10.in-addr.arpa)
Parsing zone dynamic.com (db.dynamic.com)
Parsing zone iptelco.com (db.iptelco.com) Storing named configuration
The "allow-update" config option has been set on a static zone.
This is an unrecommended (but legal)
configuration that could cause possible loss of data.
Do you want to continue? [yes] > yes

Storing records from zone 10.in-addr.arpa
...importing records into existing zone [MasterZone 10.in-addr.arpa]
...stored 9 records succesfully
Storing records from zone dynamic.com
...importing records into existing zone [MasterZone dynamic.com]
Would you like to create a PTR Record for the assignment
of the address "10.170.4.30" to the dns name "test1.dynamic.com"
(it will be added to zone "10.in-addr.arpa")? [yes] >no!!
```

Two messages are prompted in the example above. The first message occurs when a zone is imported which had previously configured to allow dynamic updates. Type **yes** to continue and all zones will be set to static. In the second message, type **no** to avoid creating duplicated *PTRRecord* objects as all the *PTRRecord* objects have been recorded in the zone files and will be loaded later.

2. Check the *SOARecord* in the DNS zone file.

```
# vi /var/named/db.iptelco.com
.....
$TTL 86400
iptelco.com. IN SOA dns1.iptelco.com.
root.dns1.iptelco.com. 3 10800 3600 604800 86400
.....
```



Find the name of *SOARecord* in the text file. In this example, the name of *SOARecord* is `dns1.iptelco.com`, and this is also the `dnsname` of the DNS server.

3. Configure DNS server using the following command:

```
IPWorks> list dnsserver
[DnsServer dns1]
  Partition: active
  Name: dns1
  Filename: named.conf
  AlgServerType: false
  ExportNeeded: true
  Option: directory "/var/named"

IPWorks> modify dnsserver dns1 -set
address=10.170.4.30;dnsname=dns1.iptelco.com -remove
option="directory \"/var/named\" -add
option="listen-on port 5300 {any;}"
1 object(s) were updated.
```

The `address` is the IP address of the machine that holds the DNS server.

The `dnsname` is the name of the DNS server. It must be the same as the name of *SOARecord*.

Remove option `directory` as it is not needed.

If there is ENUM server, add option `listen-on`.

4. Review and configure the *MasterZone*:

```
IPWorks> list masterzone
[MasterZone 10.in-addr.arpa]
  Partition: active
  .....

[MasterZone dynamic.com]
  Partition: active
  .....
  IsDynamic: false
  Option: allow-query {key dns1-default-smkey;any;},
  allow-transfer {key dns1-default-smkey;any;},
  allow-update {key dns1-default-smkey;} [MasterZone iptelco.com]
  Partition: active
  .....
```

Set the zone that has the option `allow-update` to a dynamic zone using the following command, in this example, zone `dynamic.com` has the option `allow-update`.

```
IPWorks> modify masterzone dynamic.com -set isdynamic=true
Working on 1 object(s).
1 object(s) were updated.
```

5. Start DNS Server Manager:



```
#ipw-ctr start dnssm <PL hostname>
start dnssm ==> success.
```

6. Update the *dnsserver* by using IPWorks CLI:

Note: Before update the dnsserver, make sure that the DNS server is started.

```
#ipw-ctr start dns <PL hostname>
start dns ==> success.
```

```
IPWorks> update dnsserver dns1
Result of performing an export is:
Exported the zone [MasterZone 10.in-addr.arpa]
Exported the zone [MasterZone dynamic.com]
Activated the zone [MasterZone iptelco.com]
Exported configuration for [DnsServer dns1]
Updated the configuration and started 'DNS
' server 'dns1'.
```

Example 2 Import DNS Configuration to a previously created DNS Server

The example below shows the import operation procedure that imports a DNS configuration into a DNS server that has been previously created.

Suppose the DNS server *dns1* has been previously created:

```
IPWorks> list dnsserver
[DnsServer dns1]
Partition: active
Name: dns1Address: 10.170.4.30
PrimaryAddress: 10.170.4.30
DnsName: dns1.example.com
PrimaryDnsName: dns1.example.com
Filename: named.conf
AlgServerType: falseStatus: On 07/22/08 at 18:14:14 server is 'down'
ExportNeeded: true
```

1. Check the *SOARecord* in the DNS zone file.

```
# vi /var/named/db.iptelco.com
.....
$TTL 86400
iptelco.com. IN SOA dns1.iptelco.com. root.dns1.iptelco.com. 3 10800 3600 604800 86400
.....
```

Find the name of *SOARecord* in the text file. In this example, the name of *SOARecord* is *dns1.iptelco.com*, and this is also the *dnsname* of the DNS server.

2. Configure the DNS server *dns1* to set the *dnsname* as the same as the name of *SOARecord*:



```
IPWorks> modify dnsserver dns1 -set dnsname=dns1.ipstelco.com
-add option="listen-on port 5300 {any;}"
Working on 1 object(s).
1 object(s) were updated.
```

3. Follow Step 1 to Step 6 in Example 1 to complete the import operation.

3.4.7.2 Importing DNS Zone Files

In addition to the import DNS command, the CLI has a second command that can import a single DNS zonefile from a BIND server. This command is `import zone` and the qualifiers on the command are similar to the qualifiers on the `import dns` command. For specific details, consult the online help in the CLI for this command.

The `import zone` command can import new resource records into a zone that already has resource records in it, but it cannot reimport or overwrite resource records that already exist. If imported resource records already exist, the importer produces errors.

Note: If the *IncludeRecord* and *IncludeFile* are configured in a zone, ensure that SS is able to access the included file to be imported.

3.4.7.3 Exporting Configuration Files

Many of the configuration objects managed with IPWorks can be displayed by using their configuration file format. It can export the configuration of a single object (and in some cases the configuration objects it contains).

The following example shows how the configuration of a zonefile is displayed to the terminal. This can be redirected to an external file using the `-output` qualifier and that would effectively export the configuration of the zone.



```
IPWorks> select masterzone example.com
Selected 1 object(s).
IPWorks> list -format=conf
zone "example.com" IN {
    type master;
    file "qatest_com.hosts";
    allow-query { any; };
    allow-transfer { any; };
};
IPWorks> list -format=zonefile
; Zone file for "example.com" in "view1" view for
; server "cousy".
;
; This file was generated by IPWorks at 06/06/06 16:31:54
;
; It should not be edited directly because changes
; will be overwritten by IPWorks whenever the server's
; configuration is updated in IPWorks.
;
$TTL 86400

example.com.      IN SOA cousy.example.com. postmaster. 4 1
0800 3600 604800 86400

; NS Records for authoritative servers:
example.com.      IN NS cousy.example.com.

; Resource Records:

cousy.example.com.      IN A 10.5.0.28
laura.example.com.      300 IN A 10.5.1.1
www.example.com.        IN A 10.5.1.1
www.example.com.        IN A 10.5.1.2
www.example.com.        IN A 10.5.1.3
www.example.com.        IN A 10.5.10.1
www.example.com.        IN A 10.5.10.2
www.example.com.        IN A 10.5.10.3
```

The objects that represent actively managed processes on the network (*dnsservers*, *monitors*) also support an export operation. The export operation can export all their configuration files to the local file system.

This example shows the use of the `export` command to export the configuration for a DNS Server. This exports all the files for the server to the specified directory.

```
IPWorks> select dnsserver cousy
Selected 1 object(s).
IPWorks> export -dir=/test/exp
Exported the zone [LocalhostZone cousyview1localhost]
Exported the zone [LoopbackZone
cousyview10.0.127.in-addr.arpa]
Exported the zone [MasterZone .]
Exported the zone [MasterZone 10.in-addr.arpa]
Exported the zone [MasterZone 11.in-addr.arpa]
Exported the zone [MasterZone example.com]
Exported the zone [MasterZone example.com]
Exported configuration for [DnsServer cousy]
```



3.4.7.4 Importing and Exporting XML

The CLI supports reading and writing the configuration object using XML. The following example shows the use of an alternative output format for printing an object:

```
IPWorks> list dnsserver ns0 -format=xml
<object class="dnsserver">
  <field name="Partition">
    <value>active</value>
  </field>
  <field name="Name">
    <value>ns0</value>
  </field>
  <field name="Address">
    <value>10.0.0.1</value>
    <value>10.1.0.1</value>
    <value>10.2.0.1</value>
  </field>
  <field name="PrimaryAddress">
    <value>10.0.0.1</value>
  </field>
  <field name="DnsName">
    <value>ns0.example.com</value>
  </field>
  <field name="PrimaryDnsName">
    <value>ns0.example.com</value>
  </field>
  <field name="Filename">
    <value>named.conf</value>
  </field>
</object>
IPWorks>
```

The example includes a multi-valued field and displays how those fields are handled.

The `import xml` command imports XML descriptions of objects. The importer can overwrite and add data for existing objects, as well as create a new objects.

```
IPWorks> import xml /export/arecord.xml
```

Contents of the example files `/export/arecord.xml`:



```

<start>
<object class="ARecord">
  <field name="Partition">
    <value>active</value>
  </field>
  <field name="Container">
    <value>default</value>
  </field>
  <field name="DnsName">
    <value>sh1.example.com</value>
  </field>
  <field name="Type">
    <value>A</value>
  </field>
  <field name="Address">
    <value>10.170.1.88</value>
  </field>
</object>
<object class="arecord">
  <field name="Partition">
    <value>active</value>
  </field>
  <field name="Container">
    <value>default</value>
  </field>
  <field name="DnsName">
    <value>sh2.example.com</value>
  </field>
  <field name="Type">
    <value>A</value>
  </field>
  <field name="Address">
    <value>10.170.1.168</value>
  </field>
</object>
<object class="ARecord">
  <field name="Partition">
    <value>active</value>
  </field>
  <field name="Container">
    <value>default</value>
  </field>
  <field name="DnsName">
    <value>sh3.example.com</value>
  </field>
  <field name="Type">
    <value>A</value>
  </field>
  <field name="Address">
    <value>10.170.1.188</value>
  </field>
</object>
</start>

```

Since any object can be written using this format as well as imported, it provides a convenient interchange format for transferring data from one installation to another.

3.4.7.5

Importing Text

The IPWorks CLI also supports importing objects that are represented as a formatted text file. To use this feature, the following rules must apply to the formatted text:

1. The text file must be formatted so that there is at most one object per line in the file.



2. The order and format of the fields for the objects must be the same for every line in the file.

In many cases, the data that is exported from either a database or a spreadsheet can be represented in a format that is consistent with the rules above. The data in a database or spreadsheet can be imported into IPWorks using the `import text` command in the CLI. This importer can overwrite and add data for existing objects, as well as create new objects.

Two examples are shown as follows:

Example: Import objects *ARecord* from external file `/export/arecord.txt`

```
IPWorks> import text /export/arecord.txt -class  
s=arecord -delim=" "
```

Contents in file `/export/arecord.txt`:

```
Partition Container DnsName Type Address  
active default sh188.example.com A 10.170.1.188  
active default sh88.example.com A 10.170.1.88  
active default sh8.example.com A 10.170.1.8
```

Example: Import objects *pool* from external file `/export/pool.txt`

```
IPWorks> import text /export/pool.txt -class=  
pool -delim=" "
```

Contents in file `/export/pool.txt`:

```
Partition Name Subnet AddressRange Server Area  
active Provisioned1 subnet_170 10.170.3.1-10.170.3.4  
dhcp_prim default  
active Provisioned2 subnet_170 10.170.3.9-10.170.3.15  
dhcp_prim default
```

For details on using this command, consult the online help available in the CLI.

3.5 ASDNS Operation

This section describes common operation instructions for the ASDNS server. The following ASDNS-related operations can be performed by the user:

- Starting and Stopping a Monitor, see Section 3.5.1 on page 58
- Updating Monitors, see Section 3.5.2 on page 59
- Monitoring Load, see Section 3.5.3 on page 61
- Monitoring ePDG Node, see Section 3.5.4 on page 63



3.5.1 Starting and Stopping a Monitor

IPWorks provides mechanisms for controlling a monitor when it is installed and operating. Users can use `ipw-ctr` command to start or stop the monitor directly from the system where the server is in operation.

Users can also start and stop monitors after the monitors are operational. It is required that the Server Manager for that monitor to be running, otherwise the monitor will not start successfully.

The following example shows to how to start and stop an ASDNS monitor:

1. Log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>
Password:<Password>
```

2. Start ASDNS Monitor.

- a. Start ASDNS Monitor Server Manager.

```
#ipw-ctr start asdnssm <PL hostname>
Start asdnssm ==> success.
```

- b. Start ASDNS Monitor.

```
#ipw-ctr start asdns <PL hostname>
Start asdns ==> success.
```

3. Stop ASDNS Monitor.

- a. Stop ASDNS Monitor.

```
#ipw-ctr stop asdns <PL hostname>
Stop asdns ==> success.
```

- b. Stop ASDNS Monitor Server Manager.

```
#ipw-ctr stop asdnssm <PL hostname>
Stop asdnssm ==> success.
```

3.5.2 Updating Monitors

When a monitor is configured using the IPWorks CLI, the changes will take effect in the network only after they are activated by updating the monitor. It is allowed to roll back changes, or prepare a set of changes to activate. However, it is also required that the monitor is updated once the changes are made.

See the following commands for an example of updating a single monitor:

```
IPWorks> import xml /opt/ipworks/ss/db/install.xml
```



```
IPWorks> select monitor testmon
Selected 1 object(s).
IPWorks>update
Result of performing an export is:
  Exported configuration for [Monitor testmon]
  Updated the configuration and started 'ASDNSM
ON' server 'testmon'.
```

Note:

- For the command `import xml /opt/ipworks/ss/db/install.xml`, it needs to be executed if you update the monitor action after upgrading system.

If the IPWorks system is reinstalled, this command is not necessary to be executed.

- This output message is displayed only when the Server Manager is started and `update` is executed. If the `update` command is executed when the server is running, the following message is displayed:

```
Updated the configuration for 'ASDNSMON' server
'testmon'
```

For an example of configuring monitors, follow the steps below to create `asdnsmmon.conf` file in the `/etc/ipworks/<PL hostname>/asdnsmmon` directory:

1. `IPWorks> create asdnssite internalarea EastLAN -set address=10.0.0.1/16`
2. `IPWorks> create asdnspolicy internalarea PreferLocalApns -set prefer="10.0.0.1/16 prefer EastLAN"`
3. `IPWorks> create asdnsrecord internalarea Internet.mnc001.mcc001.gprs -set policy=PreferLocalApns`
4. `IPWorks> update dnsserver`

The above steps create `asdns.conf` file in the `/etc/ipworks/<PL hostname>/dns` directory.

5. `IPWorks> create monitor indns1mon -set dnsname=indns1.example.net -set address=10.0.0.1`
6. `IPWorks> create monitormethod pingevery5minutes -set interval=5m -set filename=pingmonitor -set argument=@address`
7. `IPWorks> create monitorresource indns1 ggsn1 -set address=10.1.0.2 -set method=pingevery5minutes -set reportcontact=indns1`



8. IPWorks> **update monitor**

3.5.3

Monitoring Load

Each site contains a number of host nodes (corresponding to individual IP addresses) which are found by matching *ARecords* in the zone information for the balanced domain name with the netmasks specified in the site.

During a query on a domain, the resource list is searched for the corresponding domain name and the preferred site and host. The search for site and host are based on the relative loads.

Both the site and the node hold load information which is updated at regular intervals. When an ASDNS monitor sends an update to the DNS Server the load for all nodes in the site and the site load is calculated.

The load information consists of the aggregate backend load for each site and the specific load of each frontend node in the site. The aggregate backend load is used to determine which site should be preferred, that is, preference is given to the site which has least load. Preference is given to the front end (host) which has the least load within the site. The node chosen is then listed first in the response followed by other available front ends within that site.

For each node and site, a cumulative proportion of traffic is determined based on the updated load information. During a query, a random number in the range [0,1] is determined and the site or node with the cumulative proportion range in which the random number falls is chosen.

When the monitor updates the DNS Server, it sends two values, the frontend value and the backend value, to the DNS Server.

In the monitor, the values are calculated by the following the equation:

Backend value = threshold / load_from_script

With:

- threshold is configured in the monitor.
- load_from_script is the value which is sent from the script.
- frontend value = 0.97 if backend value <= 1
- frontend value = 1.03 if backend value > 1

See the following commands for a log-file generated from DNS Server could look like this:

```
Load info received 153.88.204.149 LB 2.0 4 1.030000 1.020408 *":
finding, node 153.88.204.149
node 153.88.204.149:Accepting load info from 153.88.204.149 asinfo msg name
example.com ip 153.88.204.149 1.030000 1.020408 status 4
```



Information is sent from the monitor and received in the DNS Server.

```
153.88.204.149  Ipaddress monitored
LB 2.0          AS-DNS version in IPWorks4.1
fe 0.97  fe means front end with value 0.97
4        Status = 4, means monitored machine is up.
          Status = 2, means monitored machine is down.
          No more reports are sent to dns until monitored
          machine comes up.
          Status = 3, monitored machine has come up after
          being reported up.
be 1.020480  be means back end
          The threshold has been set to 1.0 for this
          configuration
          Load from monitoring was 0.98
          be = threshold/load = 1.0/0.98 = 1.020408
*          domain
```

This data is handled in the following way in the DNS Server.

For raw backend load values, perform backend load averaging for site balancing.

```
node load 1.020408
node load 1.000000
average site load 1.010204 across 2 nodes
```

The new load on the node is calculated like this:

```
load_on_node = load_on_node * fe
```

fe is 0.97 in this example and load_on_node is the load that was calculated last time when the monitor sent an update for this IP address.

Summarize the total load for all nodes that are part of this site.

The load node is calculated as:

```
new_load = load / total_load
```

The fraction is calculated as:

```
fraction = total_load - (load_on_node / total_load)
```

Program wise a routine is called recursively that is why the fraction is printed.

Normalize the load across all the nodes that are up in a site for node selection during a query.

In this case, there are two nodes.

```
total load 0.986555
```



```
node norm name <dnsname> ip 153.88.204.151 load 0.448791
  frac 0.544892
node norm name <dnsname> ip 153.88.204.149 load 0.537329
  frac -0.000000
```

Now all nodes in a site have been recalculated.

The site must be recalculated next.

```
total site load 1.010000
site norm load 1.000000 frac 0.000000 cong 0.000000
```

Calculate congestion for the site.

The backend value that was sent from the monitor is not used directly. It is used to calculate average site load first, and then calculate a new be load for the following `site_congestion` and `site-load`.

```
average site load 1.010204 across 2 nodes
```

If the average site has a total load > 1.0, set backend to 1.02.

If the average site has a total load <= 1.0, set backend to 0.98.

```
if be < 1.0
    site_congestion = site_congestion +
    (0.05/number_of_nodes_that_are_up_in_the_site)
else
    site_congestion = site_congestion -
    ((0.05*0.25)/number_of_nodes_that_are_up_in_the_site)
```

So `site_congestion` takes 5 times as long to step down compared to increasing the value.

The new site load is calculated as follows:

```
site_load = site_load *
(1.0 + ((be - 1.0)/number_of_nodes_that_are_up_in_the_site))
```

be is either 0.98 or 1.02, see above.

The sites are normalized in the same way as was done for nodes.

If a query is made to the DNS, a random number is generated which chooses the site depending on the load and an answer is given.

3.5.4 Monitoring ePDG Node

IPWorks provides an example script `epdgmonitor`, which reports ePDG nodes status to DNS. The ePDG monitor reports the status of each ePDG node to ASDNS in the defined time interval.



If the following three conditions are all met, the node is reported as UP, otherwise the node is reported as DOWN.

- ePDG node information can be fetched on Route Processor (RP) via SNMP interface.
- The value of `eriEpdgStatusCode` is not 0.
- The value of ePDG load is lower than threshold.

To configure the `epdgmonitor` script:

1. Open the file `/etc/ipworks/<PL hostname>/asdnsmon/Weight.xml`.

```
# vi /etc/ipworks/<PL hostname>/asdnsmon/Weight.xml
```

2. Configure the file according to the following rules. See for an example after a successful configuration.

- For each factor section, the weight value is mandatory, and multiplier item is optional (its default value is 1). The number of entry items in each factor section is one or two.
- The value of each entry item must be one item of `EriEpdgLoadStatusEntry` defined in `ERICSSON-EPDG-LOAD-MIB.mib`. Otherwise the value is not valid.
- If the node status is marked as UP, the script returns an exit status of zero. Otherwise the script returns a non-zero exit status.
- The load is calculated by a series of parameters, such as, `eriEpdgCpuUsage`, `eriEpdgSessions`, `eriEpdgCeps`, and `eriEpdgMemoryUsage`. The value of load is calculated as following:
 - For every factor section, if the number of entry item is one, the value is this entry. If the number of entry items is two, the value is $(\text{entry 1} \div \text{entry2})$. And in the factor item, $(\text{final value}) = (\text{value}) * (\text{multiplier})$. Multiplier is optional, the default is 1.
 - $\text{Load} = \text{Sum of every factor's (final value)} * (\text{weight})$.

An example after configuration is displayed as follows:

In the example, The value of the load is:

`eriEpdgCpuUsage*0.01*0.1+(eriEpdgSessions/eriEpdgMaxSessions)*0.2+(eriEpdgCeps/eriEpdgMaxCeps)*0.3+eriEpdgMemoryUsage*0.01*0.4`

The threshold is 0.8, so if $\text{load} \leq 0.8$, the node is marked as UP, otherwise it is marked as DOWN.

```
<?xml version="1.0" encoding="UTF-8"?>
```



```
<factors threshold = "0.8">
<factor multiplier = "0.01" weight = "0.1">
<entry>eriEpdgCpuUsage</entry>
</factor>

<factor weight = "0.2">
<entry>eriEpdgSessions</entry>
<entry>eriEpdgMaxSessions</entry>
</factor>

<factor weight = "0.3">
<entry>eriEpdgCeps</entry>
<entry>eriEpdgMaxCeps</entry>
</factor>

<factor multiplier = "0.01" weight = "0.4">
<entry>eriEpdgMemoryUsage</entry>\
</factor>
</factors>
```





4 Configuring ENUM

This section guides the configuration personnel how to configure ENUM in IPWorks.

Note: If **Geographic Redundancy** is enabled, some ENUM objects only need to be configured in one site and they will be replicated automatically to another site, while other ENUM objects must be configured in both sites.

For more detail, refer to section *Redundancy with Single Provisioning* in *IPWorks Geographic Redundancy* to get more detail.

4.1 Configuring ENUM

This section describes common configuration instructions of the ENUM server.

The configuration of ENUM server contains the following topics:

- Listing EnumServer, see Section 4.1.1 on page 68
- Configuring EnumZone, see Section 4.1.2 on page 68
- Creating EnumAcl, see Section 4.1.3 on page 70
- Creating EnumView, see Section 4.1.4 on page 71
- Configuring EnumZVRel, see Section 4.1.5 on page 71
- Creating EnumSOARecord, see Section 4.1.6 on page 72
- Creating EnumNSRecord, see Section 4.1.7 on page 72
- Configuring EnumDnSched, see Section 4.1.8 on page 73
- Creating AINNNode, see Section 4.1.9 on page 78
- Creating AINLNpData, see Section 4.1.10 on page 78
- Creating DestNode, see Section 4.1.11 on page 78
- Configuring EnumDnRange, see Section 4.1.12 on page 79
- Creating AINTollFreeData, see Section 4.1.13 on page 82
- Creating CountryCode, see Section 4.1.14 on page 83
- Creating INAPData, see Section 4.1.15 on page 83
- Creating INAPNode, see Section 4.1.16 on page 83



- Creating MAPData, see Section 4.1.17 on page 83
- Creating MAPNode, see Section 4.1.18 on page 84
- Creating SCCPAddress, see Section 4.1.19 on page 84
- Configuring EnumOperator, see Section 4.1.20 on page 84
- Configuring EnumClientRealm, see Section 4.1.21 on page 87

For more information about the ENUM objects, refer to the *ENUM Objects* section in *IPWorks DNS, ASDNS, ENUM Parameter Description*.

4.1.1 Checking Enumserver Information

The *enumserver* objects are created by IPWorks. The object can only be modified in IPWorks CLI. To check the information of the objects, use the following command:

```
IPWorks> list enumserver
[[EnumServer 1]
  enumServerId: 1
  defaultMTU: 1500
  defaultNaptrOrder: 100
```

Example 3 Checking the enumserver object in IPWCLI

To check the information of *EnumServer* MO in ECLI, do the following:

```
>show -v ManagedElement=<Node Name>,IpworksFunction=1,IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1
EnumServer=1
  dbConnectString="SC-1:1186" <default>
  dbConnectStringSecondary="SC-2:1186" <default>
  dnsResolver=true <default>
  dnsResolverIPAddress="127.0.0.1" <default> <read-only>
  dnsResolverPort=5300 <default>
  enumServerId="1" <default>
  ipv4Address="0.0.0.0" <default>
  ipv6Address="::" <default>
  port=53 <default>
  securitylog=false <default>
  threadCount=50 <default>
  Erh=1
  Log=1
```

Example 4 Checking the EnumServer MO in ECLI



4.1.2 Configuring EnumZone

This section describes how to create and delete an ENUM Zone.

4.1.2.1 Creating Enumzone

To create an *EnumZone* object, use the following command.

```
IPWorks> create enumzone <ENUM Zone Id> -set enum
zonename=<ENUM Zone Name>
```

For example:

```
IPWorks> create enumzone 1 -set enumzonename=
"e164.iptelco.com"
```

4.1.2.1.1 Creating Subzone

To create an ENUM Zone that is a subzone of a larger ENUM Zone (for example, 6.4.e164.iptelco.com to e164.iptelco.com), follow the procedure below so that the related enumdnsched records can be transferred to the new subzone.

1. Stop the Storage Server.

```
# ipw-ctr stop ss <SC hostname>
stop ss ==> success.
```

2. Create a new subzone using the following command:

```
# /opt/ipworks/mysql/scripts/ipwenum_zonemgm.pl
create_subzone <Enum Zone Id> <Enum Zone Name>
```

Example:

```
# /opt/ipworks/mysql/scripts/ipwenum_zonemgm.pl
create_subzone 2 6.4.e164.iptelco.com
```

The following outputs are displayed:

```
There is(are) 1 ENUMDNSCHED record(s) will be
moved from e164.iptelco.com to 6.4.e164.iptelco.com
Maybe it will take a long time. Do you wish to
continue?(default: Yes) [Yes/No]::
```

3. Type **Yes** and press **Enter** to confirm the process. When the process finishes, the following texts are displayed:

```
Create ENUMZONE [ 2 6.4.e164.iptelco.com ] successfully
Start moving ENUMDNSCHED from e164.iptelco.com to
6.4.e164.iptelco.com
```



```
There is(are) 1 enumdnsched record(s) has(have) been
moved to 6.4.e164.iptelco.com successfully
```

4. Start the Storage Server.

```
# ipw-ctr start ss
start ss ==> success.
```

4.1.2.2 Deleting Enumzone

To delete an ENUM Zone that contains one or more *enumdnsched* records, perform the following steps:

1. Stop the Storage Server.

```
# ipw-ctr stop ss <SC hostname>
stop ss ==> success.
```

2. Delete the desired ENUM Zone by executing the following command:

```
# /opt/ipworks/mysql/scripts/ipwenum_zonemgm.pl
delete_subzone <Enum Zone Id>
```

After the *enumzone* is deleted, the related *enumdnsched* records are transferred to the parent *enumzone*; if no parent *enumzone* exists, all the *enumdnsched* records are deleted.

Example:

```
# /opt/ipworks/mysql/scripts/ipwenum_zonemgm.pl
delete_subzone 2
```

After the command is executed, the following messages are displayed:

```
There is(are) 1 ENUMDNSCHED record(s) will be moved
from 6.4.e164.iptelco.com to e164.iptelco.com
Maybe it will take a long time. Do you wish to
continue?(default: Yes) [Yes/No]::
```

3. Type **yes** and press **Enter** to confirm the process.

After the process finishes, the following messages are displayed:

```
Start moving ENUMDNSCHED from 6.4.e164.iptelco.com to
e164.iptelco.com
There is(are) 1 ENUMDNSCHED record(s) has(have) been
moved to e164.iptelco.com successfully
Delete ENUMZONE [ 2 6.4.e164.iptelco.com ] successfully
```

4. Start the Storage Server.

```
# ipw-ctr start ss
start ss ==> success.
```



4.1.3 Creating EnumAcl

To create an *EnumAcl* object, use the following command.

```
IPWorks> create enumacl <Acl Id> -set aclname=<Acl
Name>;matchlist="<Match List>"
```

For example:

```
IPWorks> create enumacl 1 -set aclname=acl1;matchl
ist="{10.0.0.1; 10.170.0.0/16;}"
```

4.1.4 Creating EnumView

To create an *EnumView* object, use the following command.

```
IPWorks> create enumview <View Id> -set viewn
ame=<View Name>;
rank=<Rank>;aclid=<Acl Id>;serveridlist="<Server Id List>"
```

For example:

```
IPWorks> create enumview 1 -set viewname=view1;ran
k=100;aclid=1;serveridlist="1"
```

4.1.5 Configuring EnumZVRel

This section describes how to create and modify an ENUM Zone View Relationship.

4.1.5.1 Creating EnumZVRel

To create an *EnumZVRel* object, use the following command.

```
IPWorks> create enumzvrel -set zoneid=<Zone
Id>;viewid=<View Id>
```

For example:

```
IPWorks> create enumzvrel -set zoneid=1;viewid=1
or (alternative format)
IPWorks> create enumzvrel 1 1
```

4.1.5.2 Modifying EnumZVRel

Assume that the view association of *zone 1* needs to be changed from *view 1* to *view 2*.



1. If *zone 1* only contains the *EnumDnSched* or nothing, modify the *viewid* value of the *EnumZVRel*.

```
IPWorks> modify enumzvrel 1 1 -set viewid=2
```

2. If *zone 1* contains any *EnumDnRange* that is associated with *view 1*, it is not allowed to modify the *viewid* value of the *EnumZVRel* directly. Apply the following method instead.

- a. Create a new *EnumZVRel*.

```
IPWorks> create enumzvrel -set zoneid=1;viewid=2
```

- b. Modify the *viewid* value of all the *EnumDnRange* in *zone 1*.

```
IPWorks> modify enumdnrange -where enumzon  
eid=1&viewid=1 -set viewid=2
```

- c. Delete the old *EnumZVRel*.

```
IPWorks> delete enumzvrel 1 1
```

4.1.6 Creating EnumSOARecord

To create an *EnumSOARecord* object, use the following command:

```
IPWorks> create enumsoarecord <Domain Name> -set  
serverid=<ENUM Server Id>;nameserver=<DNS Name>;  
minimum=<Number>
```

For example:

```
IPWorks> create enumsoarecord e164.arpa -set serverid  
=1;nameserver=example.com;minimum=86400
```

Note: If *enumserver* has defined its *dnsname*, when creating *enumzone*, the corresponding *enumsoarecord* will be created automatically.

4.1.7 Creating EnumNSRecord

To create an *EnumNSRecord* object, use the following command:

```
IPWorks> create enumnsrecord <Domain Name> -set  
serverid=<ENUM Server Id>;nameserver=<DNS Name>
```

For example:

```
IPWorks> create enumnsrecord e164.arpa -set serve  
rid=1;nameserver=example.com
```



Note: If *enumserver* has defined its *dnsname*, when creating *enumzone*, the corresponding *enumnsrecord* will be created automatically.

4.1.8 Configuring EnumDnSched

This section describes how to create, delete, and list *EnumDnSched*.

4.1.8.1 Creating EnumDnSched

```
IPWorks> create enumdnsched <Enum Dn> -set
naptrFlags=<NAPTR Flags>;naptrOrder=<NAPTR Order>;
naptrPreference=<NAPTR Preference>;naptrservice=<NAPTR Service>;
naptrTxt=<NAPTR Txt>;propblocking=<Probably Blocking>
```

The following formats of *EnumDnSched* are supported:

- Reverse dotted notation format

```
IPWorks> create enumdnsched 4.3.2.1.4.3.2.
1.0.1.4.4.e164.
iptelco.com -set naptrFlags=n;naptrOrder=100;
naptrPreference=10;naptrservice=E2U+SIP;naptrTxt=!^.*$!sip:441012341234@iptelco.com!;propblocking=0
```

For this format, an MSISDN, such as +441012341234, is provisioned as 4.3.2.1.4.3.2.1.0.1.4.4.e164.iptelco.com where it will be placed in the best matched enumzone.

- Contiguous number notation format

```
IPWorks> create enumdnsched 441012341234.e164.
iptelco.com -set naptrFlags=n;naptrOrder=100;
naptrPreference=10;naptrservice=E2U+SIP;naptrTxt=!^.*$!sip:441012341234@iptelco.com!;propblocking=0
```

For this format, an MSISDN, such as +441012341234, is provisioned as 441012341234.e164.iptelco.com where it will be only placed in the enumzone e164.iptelco.com. If the enumzone e164.iptelco.com is not configured, the CLI command will be rejected.

When using contiguous number notation format, the user must know the best matched zone when creating an *EnumDnSched* object. Otherwise, the CLI command will be rejected. For example, if the user wants to create an *EnumDnSched* object 441012341234.e164.iptelco.com when the existing best matched zone is 3.2.1.4.3.2.1.0.1.4.4.e164.iptelco.com, the CLI command to create such an object is rejected because the contiguous number notation format considers e164.iptelco.com as the best matched zone that differs from the real best matched zone.

Use one of the following commands to view the created *EnumDnSched* object:



```
IPWorks> list enumdnsched 4.3.2.1.4.3.2.1.0.1.  
4.4.e164.iptelco.com  
Or  
IPWorks> list enumdnsched 441012341234.e164.iptelco.com
```

4.1.8.2 Deleting EnumDnSched

Use one of the following commands to delete the *EnumDnSched* object:

```
IPWorks> delete enumdnsched 3.2.1.4.4.e164.iptelco.com  
Or  
IPWorks> delete enumdnsched 44123.e164.iptelco.com
```

After the command is executed successfully, the output is displayed as below:

```
Working on 1 object(s).  
1 object(s) were updated.
```

Example:

1. List *EnumDnSched* to get some information needed.

```
list enumdnsched  
[EnumDnSched 1 44123]  
  enumDn: 44123  
  enumZoneId: 1  
  propBlocking: 0  
  naptrOrder: 12  
  naptrPreference: 10  
  naptrService: E2U+SIP  
  naptrFlags: nU  
  naptrTxt: /^.*$/sip:86005000040828-1@ipworks.ims.net/  
  naptrOrder2: 13  
  naptrPreference2: 20  
  naptrService2: E2U+SIP  
  naptrFlags2: nU  
  naptrTxt2: /^.*$/sip:86005000040828-2@ipworks.ims.net/  
  naptrOrder3: 14  
  naptrPreference3: 30  
  naptrService3: E2U+SIP  
  naptrFlags3: nU  
  naptrTxt3: /^.*$/sip:86005000040828-3@ipworks.ims.net/  
  naptrOrder4: 15  
  naptrPreference4: 40  
  naptrService4: E2U+SIP  
  naptrFlags4: nU  
  naptrTxt4: /^.*$/sip:86005000040828-4@ipworks.ims.net/  
  naptrOrder5: 16  
  naptrPreference5: 50  
  naptrService5: E2U+SIP  
  naptrFlags5: nU
```



```
naptrTxt5: /^.*$/sip:86005000040828-5@ipworks.ims.net/
```

2. Record enumZoneId: 1 and enumDn: 44123.
3. Get the enumZoneName according to the enumZoneId:

```
IPWorks> list enumzone 1
[EnumZone 1]
  enumZoneId: 1
  enumZoneName: e164.ipstelco.com
  InDefaultView: true
```

4. Combine enumDn: 44123 with enumZoneName: e164.ipstelco.com, and delete the EnumDnSched object.

```
IPWorks> delete enumdnsched 44123.e164.ipstelco.com
Working on 1 object(s).
1 object(s) were updated.
```

4.1.8.3 Listing EnumDnSched

This section provides the information on how to list the *EnumDnSched* object.

4.1.8.3.1 Setting List Preference for EnumDnSched

Storage Server usually stores millions of the *EnumDnSched* objects. Listing them all might overwhelm the screen and affect the performance of the server. Therefore `CLI.DNLimit`, which is a limitation of the number of records that can be displayed on the screen, is introduced. Its default value is 200, and the valid value range is from 1 to 1000.

The user now can set the limitation for the following scenarios:

- *User*: The value is valid as long as the user is logged on with the same user account.

```
IPWorks> set CLI.DNLimit=100 -save
```

- *Session*: The value is only valid in the current session (it expires once the user logs out of the CLI).

```
IPWorks> set CLI.DNLimit=100
```

- *Command*: The value is valid only once when the command is executed.

```
IPWorks> list enumdnsched -preference="CLI.DNLimit=100"
```

The limitation can be displayed by using the following command:



```
IPWorks> show preference
CLI.DNLimit=100
```

Example 5 Showing the Display Limitation of Records Number

When listing EnumDnSched in CLI, only the records within the limitation are displayed. If the number of records exceeds the limitation, the user is prompted whether to dump all the records to a file.

```
IPWorks> list enumdnsched
[EnumDnSched 1 44123]
...
[EnumDnSched 1 44323]
  enumDn: 44323
  enumZoneId: 1
  propBlocking: 0
  naptrOrder: 12
  naptrPreference: 10
  naptrService: E2U+SIP
  naptrFlags: nU
  naptrTxt: /^.*$/sip:86005000040828-1@ipworks.ims.net/
  naptrOrder2: 13
  naptrPreference2: 20
  naptrService2: E2U+SIP
  naptrFlags2: nU
  naptrTxt2: /^.*$/sip:86005000040828-2@ipworks.ims.net/
  naptrOrder3: 14
  naptrPreference3: 30
  naptrService3: E2U+SIP
  naptrFlags3: nU
  naptrTxt3: /^.*$/sip:86005000040828-3@ipworks.ims.net/
  naptrOrder4: 15
  naptrPreference4: 40
  naptrService4: E2U+SIP
  naptrFlags4: nU
  naptrTxt4: /^.*$/sip:86005000040828-4@ipworks.ims.net/
  naptrOrder5: 16
  naptrPreference5: 50
  naptrService5: E2U+SIP
  naptrFlags5: nU
  naptrTxt5: /^.*$/sip:86005000040828-5@ipworks.ims.net/
```

There are too many records to display :1837148 records found.
Do you want to output these records to a file? [yes or y]

Example 6 Listing EnumDnSched within the Display Limitation

The path and name of the file is required when typing **yes** or **y**.

Note: If the number of the records is greater than one million, the time for dumping the file might take as long as three hours.



4.1.8.3.2 Wildcard

Wildcard is also supported when listing *EnumDnSched*. Only one * is supported. It must be the first character in reverse dotted notation format and the last character of the *EnumDn* in contiguous number notation format.

```
IPWorks> list enumdnsched *.2.1.4.4.e164.iptelco.com
```

Example 7 Using Wildcard in Reverse Dotted Notation Format

```
IPWorks> list enumdnsched 4412*.e164.iptelco.com
```

Example 8 Using Wildcard in Contiguous Number Notation Format

4.1.8.3.3 Where Qualifier

The *where* qualifier is also supported.

```
IPWorks> list enumdnsched -where enumdn=44123&enumzoneid=1
```

Example 9 Using Where Qualifier

4.1.8.3.4 Listing Dn

The *list dn* command is used to view the *EnumDnSched* and *EnumDnRange* at the same time.

For example:

```
IPWorks> list dn
***EnumDNRange***
[EnumDNRange 1 2]
  enumZoneId: 1
  enumDnRange: 2
  naptrOrder: 2
  naptrPreference: 1
  naptrService: E2U+SIP
  naptrFlags: n
  naptrTxt: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder2: 21
  naptrPreference2: 2
  naptrService2: E2U+SIP
  naptrFlags2: n
  naptrTxt2: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder3: 2
  naptrPreference3: 3
  naptrService3: E2U+SIP
  naptrFlags3: n
  naptrTxt3: !^.*$!sip:44123@e164.iptelco.com!
...
***EnumDNSched***
[EnumDnSched 1 40823]
```



```

enumDn: 40823
enumZoneId: 1
propBlocking: 0
naptrOrder2: 14
naptrPreference2: 30
naptrService2: E2U+SIP
naptrFlags2: nU
naptrTxt2: /^.*$/sip:86001000040823-3@ipworks.ims.net/
naptrOrder4: 15
naptrPreference4: 40
naptrService4: E2U+SIP
naptrFlags4: nU
naptrTxt4: /^.*$/sip:86001000040823-4@ipworks.ims.net/
...

```

4.1.9 Creating AINNode

To create an *AINNode* object, use the following command:

```

IPWorks> create AINNode -set enumserverid=<Enum Server Id>;
ainuserid=<AIN User Id>;localssn=<local SSN>;
localspc=<Local SPC>;bearercapability=<Number>;
npswitch=<enable/disable NP function>

```

For example:

```

IPWorks> create AINNode -set enumserverid=1;
ainuserid=2135476980;localssn=10;localspc=100
;bearercapability=0;npswitch=1

```

4.1.10 Creating AINLNPDData

To create an *AINLNPDData* object, use the following command:

```

IPWorks> create AINLNPDData -set ProtocolSettingName=<AIN Setting
Name>;TranslationType=<Type Number>;
ServiceParameter=<Format>;Domain=<Domain Name>;CallingPartyId=<Calling Party DN>;
ChargeNumber=<AIN Number>;JurisdictionInformation=<Number>;Trig
gerCriteriaType=<Type Number>

```

For example:

```

IPWorks> create AINLNPDData -set ProtocolSettingName=lnpData1;TranslationType=3;
ServiceParameter=E2U+pstn:sip;Domain=test.com;CallingPartyId=nn:3:np:1:pres:0:scr:0:dig:9999;
ChargeNumber=nn:1:np:5:scr:0;JurisdictionInformation=4356;TriggerCriteriaType=37

```

4.1.11 Creating DestNode

To create a *DestNode* object, use the following command:



```
IPWorks> create destnode -set destnodename=<Name>; type=<Type
Number>; ProtocolSettingName=<Name>
```

For example:

```
IPWorks> create destnode -set destnodename=destNode1
; type=7; ProtocolSettingName=lnpData1
```

4.1.12 Configuring EnumDnRange

The fields `enumZoneId` and `enumDnRange` are generated automatically from the series URI string according to the best matching rule.

For example, a user wants to create the *EnumDnRange* object `4.3.2.1.4.3.2.1.0.1.4.4.e164.ipstelco.com` or `441012341234.e164.ipstelco.com` and a sequence of *EnumZone* objects with the super-zone relationship already exist (shown as follows).

The best matched zone for the *EnumDnRange* is `3.2.1.4.3.2.1.0.1.4.4.e164.ipstelco.com` with the `enumZoneId` of 21. Therefore, the `enumZoneId` and `enumDnRange` fields of the *EnumDnRange* object are set to 21 and 4 respectively.

| EnumZoneName | EnumZoneId |
|---|------------|
| e164.ipstelco.com | 10 |
| 4.e164.ipstelco.com | 11 |
| 4.4.e164.ipstelco.com | 12 |
| 1.4.4.e164.ipstelco.com | 13 |
| 0.1.4.4.e164.ipstelco.com | 14 |
| 1.0.1.4.4.e164.ipstelco.com | 15 |
| 2.1.0.1.4.4.e164.ipstelco.com | 16 |
| 3.2.1.0.1.4.4.e164.ipstelco.com | 17 |
| 4.3.2.1.0.1.4.4.e164.ipstelco.com | 18 |
| 1.4.3.2.1.0.1.4.4.e164.ipstelco.com | 19 |
| 2.1.4.3.2.1.0.1.4.4.e164.ipstelco.com | 20 |
| 3.2.1.4.3.2.1.0.1.4.4.e164.ipstelco.com | 21 |

Like *EnumDnSched*, *EnumDnRange* supports both reverse dotted notation format and contiguous number notation format as well.



Note: When creating an *EnumDnRange* object, it is not allowed to have the same name as any existing *EnumZone*. The alternative method is to create 10 *EnumDnRange* objects.

For example, if an *EnumZone* 3.2.1.4.4.e164.ip.telco.com exists and an *EnumDnRange* 44123.e164.ip.telco.com needs to be created, create the 10 *EnumDnRange* objects as follows:

0.3.2.1.4.4.e164.ip.telco.com

1.3.2.1.4.4.e164.ip.telco.com

2.3.2.1.4.4.e164.ip.telco.com

3.3.2.1.4.4.e164.ip.telco.com

4.3.2.1.4.4.e164.ip.telco.com

5.3.2.1.4.4.e164.ip.telco.com

6.3.2.1.4.4.e164.ip.telco.com

7.3.2.1.4.4.e164.ip.telco.com

8.3.2.1.4.4.e164.ip.telco.com

9.3.2.1.4.4.e164.ip.telco.com

4.1.12.1 Creating EnumDnRange for NP

The precondition is that the *DestNode* destNode1 is created.

```
IPWorks> create enumZone 10 -set enumZoneName=e164.ip.telco.com
1 object(s) created.
IPWorks> create enumDnRange 3.2.1.4.4.e164.ip.telco.com
-set destNode=destNode1
1 object(s) created.
```

If an *EnumDnRange* is created to associate with the *EnumView* 100, follow the procedure below.

```
IPWorks> create enumzvrel -set
zoneid=10;viewid=100;description="For NP DATA"
1 object(s) created.
IPWorks> create enumDnRange 3.2.1.4.4.e164.ip.telco.com
-set viewid=100;destnode=destNode1
1 object(s) created.
```

Use the following command to view the created *EnumDnRange*:

```
IPWorks> list enumdnrange 3.2.1.4.4.e164.ip.telco.com
```



4.1.12.2 Creating EnumDnRange for NAPTRRecord

```
IPWorks> create enumZone 10 -set enumZoneName=e164.iptelco.com
1 object(s) created.
IPWorks> create enumDnRange 3.2.1.4.4.e164.iptelco.com -set
NaptrFlags=n;NaptrOrder=1;NaptrPreference=2;
NaptrService=E2U+SIP;NaptrTxt=!^.*$!sip:441234@e164.iptelco.com!
1 object(s) created.
```

Use the following command to view the created *EnumDnRange*:

```
IPWorks> list enumdnrange 3.2.1.4.4.e164.iptelco.com
```

4.1.12.3 Deleting EnumDnRange

```
IPWorks> delete enumdnrange 3.2.1.4.4.e164.iptelco.com
Working on 1 object(s).
1 object(s) were updated.
```

Example:

1. List *EnumDnRange* to get some information needed.

```
IPWorks> list enumdnrange
[EnumDnRange 10 44123]
  enumZoneId: 10
  enumDnRange: 44123
  destNode: destNode1
```

2. Record enumZoneId: 10 and enumDnRange: 44123.
3. Get the enumZoneName according to the enumZoneId.

```
IPWorks> list enumzone 10
[EnumZone 10]
  enumZoneId: 10
  enumZoneName: e164.iptelco.com
  InDefaultView: true
```

4. Combine enumDnRange: 44123 with enumZoneName: e164.iptelco.com, and delete the EnumDnRange object.

```
IPWorks> delete enumdnrange 3.2.1.4.4.e164.iptelco.com
Working on 1 object(s).
1 object(s) were updated.
```

4.1.12.4 Listing EnumDnRange

The configurations for the list preference setting, *wildcard*, where qualifier, and list dn are the same for both *EnumDnRange* and *EnumDnSched*. For details, see Section 4.1.8.3 on page 75.



Each *EnumDnRange* corresponds to a maximum of five instances. To view all the instances for a certain *EnumDnRange* at a time, the user can use the *dnfrom* and *dnto* qualifiers.

For example:

```
IPWorks> list enumdnrange -dnfrom:1 -dnto:2
[EnumDnRange 1 44123]
  enumZoneId: 1
  enumDnRange: 44123
  naptrOrder: 1
  naptrPreference: 1
  naptrService: E2U+SIP
  naptrFlags: n
  naptrTxt: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder2: 21
  naptrPreference2: 2
  naptrService2: E2U+SIP
  naptrFlags2: n
  naptrTxt2: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder3: 2
  naptrPreference3: 3
  naptrService3: E2U+SIP
  naptrFlags3: n
  naptrTxt3: !^.*$!sip:44123@e164.iptelco.com!

[EnumDnRange 1 44124]
  enumZoneId: 1
  enumDnRange: 44124
  naptrOrder: 2
  naptrPreference: 1
  naptrService: E2U+SIP
  naptrFlags: n
  naptrTxt: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder2: 2001
  naptrPreference2: 2
  naptrService2: E2U+SIP
  naptrFlags2: n
  naptrTxt2: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder3: 2001
  naptrPreference3: 3
  naptrService3: E2U+SIP
  naptrFlags3: n
  naptrTxt3: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder4: 2001
  naptrPreference4: 4
  naptrService4: E2U+SIP
  naptrFlags4: n
  naptrTxt4: !^.*$!sip:44123@e164.iptelco.com!
  naptrOrder5: 2001
  naptrPreference5: 5
  naptrService5: E2U+SIP
  naptrFlags5: n
  naptrTxt5: !^.*$!sip:44123@e164.iptelco.com!

Working on 2 object(s).
```

4.1.13 Creating AINTollFreeData

To create an *AINTollFreeData* object, use the following command:

```
IPWorks> create aintollfreedata -set protocolsettingname=<Name>;
carrier=<Carrier>;chargenumber=<Charge Number>;
chargepartystationtype=<Type Number>;lata=<Lata>;translationtype=<Type Number>
```

For example:



```
IPWorks> create aintollfreedata -set protocolsettingname=protocoltf9;
carrier=cs:1:nc:0:dig:9999;chargenumber=nn:3:np:1:pres:0:scr:0:dig:9999;
chargepartystationtype=0;lata=nn:3:np:1:pres:0:scr:0:dig:999;translationtype=11
```

4.1.14 Creating CountryCode

To create a *CountryCode* object, use the following command:

```
IPWorks> create CountryCode <Country Code>
```

For example:

```
IPWorks> create CountryCode 86
```

4.1.15 Creating INAPData

To create an *INAPData* object, use the following command:

```
IPWorks> create INAPData -set ProtocolSettingName=<Name>;
ServiceParameter=<Format>;Domain=<Domain Name>;add
ressname=<Name>
```

For example:

```
IPWorks> create INAPData -set ProtocolSettingName=inap1;
ServiceParameter=E2U+pstn:sip;Domain=test.com
;addressname=sccpaddress1
```

4.1.16 Creating INAPNode

To create an *INAPNode* object, use the following command:

```
IPWorks> create INAPNode -set EnumServerId=<EnumServer
Id>;ServiceKey=<Service Key>;
LocalSSN=<Local SSN>;LocalSPC=<Local SPC>;InternationalFormat=<Num
ber>;NPSSwitch=<Enable/Disable NP>
```

For example:

```
IPWorks> create INAPNode -set EnumServerId=1;ServiceKey=123456;
LocalSSN=10;LocalSPC=100;InternationalFormat=0;NPSSwitch=0
```

4.1.17 Creating MAPData

To a *MAPData* object, use the following command:

```
IPWorks> create MAPData -set ProtocolSettingName=<Name>;Tra
nslationType=<Type Number>;
ServiceParameter=<Format>;ReplyFormat=<Number>;SipNpDomain=<Domain Name>
```



For example:

```
IPWorks> create MAPData -set ProtocolSettingName=map1;TranslationType=3;
ServiceParameter=E2U+pstn:sip;ReplyFormat=1;SipNpDomain=test.com
```

4.1.18 Creating MAPNode

To create a *MAPNode* object, use the following command:

```
IPWorks> create MAPNode -set EnumServerId=<Enum Server Id>;GsmS
cfAddress=<gsmSCF address>;
LocalSSN=<Local SSN>;LocalSPC=<Local SPC>;InternationalFormat=<Num
ber>;NPSSwitch=<Enable/Disable NP>
```

For example:

```
IPWorks> create MAPNode -set EnumServerId=1;GsmScfAddress=tn:1:np:1:dig:123;
LocalSSN=10;LocalSPC=100;InternationalFormat=0;NPSSwitch=1
```

4.1.19 Creating SCCPAddress

To create an *SCCPAddress* object, use the following command:

```
IPWorks> create sccpaddress <Address Name> -set GlobalTitleIndicator=<Glo
balTitleIndicator>;translationtype=<Type Number>
```

For example:

```
IPWorks> create sccpaddress cdpa -set GlobalTitleIndicator=2;translationtype=123
```

4.1.20 Configuring EnumOperator

This section describes how to configure the *EnumOperator* object.

4.1.20.1 Creating EnumOperator

If the users want to create an *EnumOperator* object, perform the following steps:

1. Log on to SS server by SSH.

```
# ssh <user name>@<MIP_PROV_IP>
```

2. Enter the IPWorks CLI environment.

```
# ipwcli
```

The CLI prompt appears as *IPWorks>*, which indicates the user is in CLI working environment.



3. Create an `EnumOperator` object by using the following command:

```
IPWorks>create enumoperator <operator_name> -prompt
OperatorMCC? [next] > <MCC_value>
OperatorMNC? [next] > <MNC_value>
OperatorType? [next] > <operator_type_value>
ServiceParameter? [next] > <server_parameter_value>
Subdomain? [next] > <subdomain_value>
```

Example:

```
IPWorks> create enumoperator telefonica -prompt
OperatorMCC? [next] > 460
OperatorMNC? [next] > 002
OperatorType? [next] > 1
ServiceParameter? [next] > e2u+pstn:tel
Subdomain? [next] > ims
1 object(s) created.
```

Expected Result: An `EnumOperator` object with name `telefonica` is created.

4. (Optional) Check the `EnumOperator` object list and see whether the object with name `telefonica` is created.

For details about how to check whether the object is created successfully, see Section 4.1.20.2 on page 85.

4.1.20.2 Listing EnumOperator

If the users want to check the details about an `EnumOperator` object, perform the following steps:

1. Log on to SS server by SSH.

```
# ssh <user_name>@<MIP_PROV_IP>
```

2. Enter the IPWorks CLI environment.

```
# ipwcli
```

The CLI prompt appears as `IPWorks>`, which indicates the user is in CLI working environment.

3. List the `EnumOperator` object by using the following command:

```
IPWorks> list EnumOperator
[EnumOperator telefonica]
  OperatorName: telefonica
  OperatorMCC: 460
  OperatorMNC: 002
  OperatorType: 1
  ServiceParameter: e2u+pstn:tel
  Subdomain: ims
```



Expected result: The object with name `telefonica` appears.

4.1.20.3 Modifying EnumOperator

An *EnumOperator* object has several attributes and each attribute can be configured to different value based on proper rules. If the users want to modify some attributes to meet specific requirement, perform the following steps:

1. Log on to SS server by SSH.

```
# ssh <user name>@<MIP_PROV_IP>
```

2. Enter the IPWorks CLI environment.

```
# ipwcli
```

The CLI prompt appears as `IPWorks>`, which indicates the user is in CLI working environment.

3. Modify an *EnumOperator* object by using the following command:

```
IPWorks> modify EnumOperator <operator_name> -set <attribute_name>=<attribute_value>
```

For example: For the operator `telefonica` to modify the value of attribute `OperatorMNC` as `004`, run the following command:

```
IPWorks> modify EnumOperator telefonica -set OperatorMNC=004
```

4. (Optional) Check the *EnumOperator* object list and see for operator `telefonica`, whether the value of attribute `OperatorMNC` is changed.

For details about how to check the object, see Section 4.1.20.2 on page 85.

Expected result: For operator `telefonica`, the value of attribute `OperatorMNC` is changed to `004`.

Example:

```
[EnumOperator telefonica]
  OperatorName: telefonica
  OperatorMCC: 460
  OperatorMNC: 004
  OperatorType: 1
  ServiceParameter: e2u+pstn:tel
  Subdomain: ims
```

4.1.20.4 Deleting EnumOperator

When the users do not need a specific *EnumOperator* object, delete the object by following these steps:



1. Log on to SS server by SSH.

```
# ssh <user name>@<MIP_PROV_IP>
```

2. Enter the IPWorks CLI environment.

```
# ipwcli
```

The CLI prompt appears as `IPWorks>`, which indicates the user is in CLI working environment.

3. Delete a specific by running the following command:

```
IPWorks> delete EnumOperator <operator_name>
```

For example: To delete the previously created *EnumOperator* object with operator name `telefonica`, run command:

```
IPWorks> delete EnumOperator telefonica
```

Expected result:

```
Working on 1 object(s) .
1 object(s) were updated.
```

4. (Optional) Check the *EnumOperator* object list and see whether the object with operator name `telefonica` does not exist at all.

For details about how to check the object, see Section 4.1.20.2 on page 85.

Example:

```
<error 64 computing object keys>
Object enumoperator|telefonica has been deleted
```

4.1.21 Configuring EnumClientRealm

The users are only allowed to modify or check the details of an *EnumClientRealm* object.

Note: The users do not have permission to create and delete an *EnumClientRealm* object. If the users force to perform these two actions, a warning message pops up to remind that creation or deletion the object is forbidden.

4.1.21.1 Listing EnumClientRealm

To check the details of an *EnumClientRealm* object, list the object attributes as these steps:



1. Enter the IPWorks CLI environment in SS server by following the steps (Step 1 and Step 2) in Section 4.1.20.1 on page 84.
2. List the *EnumClientRealm* object by using the following command:

```
IPWorks> list EnumClientRealm
```

Expected result: The attribute values of the object appear. Take the following command output as example, the name of the object is *clientrealm*.

```
[EnumClientRealm clientrealm]
  LocalClients: {10.170.23.0/24;}
  NationalClients: {10.170.24.2;10.170.24.200;}
  InternationalClients: {10.170.25.2;10.170.25.200;}
```

4.1.21.2 Modifying EnumClientRealm

An *EnumClientRealm* object has several attributes and each attribute can be configured to different value based on proper rules. If the users want to modify some attribute value to meet their requirements, perform the following steps:

1. Enter the IPWorks CLI environment in SS server by following the steps (Step 1 and Step 2).
2. Modify the *EnumClientRealm* object by using the following command:

```
IPWorks>modify EnumClientRealm <EnumClientRealm_object_name> -set
<attribute_name>=<attribute_value>"
```

Example: For *EnumClientRealm* object with name *clientrealm*, to modify the value of attribute *LocalClients* to a new value, such as {10.170.23.0/24;}, run the following command:

```
IPWorks>modify EnumClientRealm clientrealm -set LocalCl
ients="{10.170.23.0/23;}"
```

3. (Optional) Check the *EnumClientRealm* object.

```
IPWorks> list EnumClientRealm
```

Expected result: The value of *LocalClients* is changed to {10.170.23.0/23;};

```
[EnumClientRealm clientrealm]
  LocalClients: {10.170.23.0/23;}
  NationalClients: {10.170.24.2;10.170.24.200;}
  InternationalClients: {10.170.25.2;10.170.25.200;}
```



4.2 Configuring ENUM Access Control

ENUM Access Control is a security mechanism for the ENUM server to control the access by blocking unauthenticated queries. If the query is allowed for the client, ENUM server sends back the response. Otherwise, the ENUM server drops and logs the query.

The ENUM View is an access control mechanism for ENUM data. One ENUM View controls the access for a defined set of ENUM Zones. An ENUM View can be applied to multiple ENUM Servers. An ENUM View controls the same set of ENUM Zones regardless of which ENUM server it belongs to. Each ENUM View is configured with a rank. The rank determines the search order of the operator defined views by the ENUM server. The rank must be unique otherwise CLI returns an error.

A default ENUM View that is not created by the operator always exists. This default view is accessible to any client. When a zone is configured for an operator defined ENUM View, it will automatically be taken out of the default view.

An ENUM ACL, which is a set of allowed and disallowed client IP addresses, is used to access to an ENUM Zone managed by the ENUM View. Each ENUM View has zero or one ENUM ACL. If one ENUM ACL is defined for an ENUM View, the ENUM ACL is always the same regardless of which ENUM server it belongs to.

Figure 3 shows the relationships with an example network configuration:

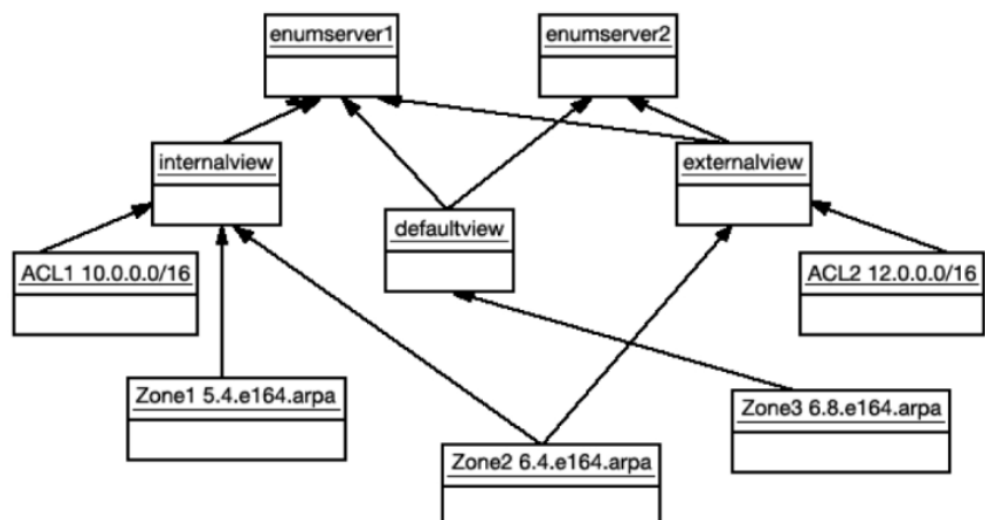


Figure 3 Relationships between ENUM Access Control Related Objects

The following steps show that how the ENUM Access Control works when the ENUM server receives a query:



1. The ENUM server finds the best matched ENUM Zone first and checks whether it is controlled by the default view.
2. If it is not controlled by the default view, the ENUM server searches related ENUM Views that the matched ENUM Zone belongs to by ascending order of the ENUM View rank value.
3. The ENUM server goes to each ENUM View in the sorted list to check the elements given in the address match list of the ENUM ACL, owned by that view until one of the following conditions occurs:
 - An element that grants the client IP address, such as "{10.170.4.28;}" or "{any;}".
 - An ENUM View without a defined ENUM ACL or an element that blocks the client IP address, such as "{!10.170.4.0/8;}", "{ }", or "{none;}".
 - Finished searching all elements in sorted ENUM Views.

Suppose that there is a sample network containing an internal and external network as shown in Figure 3. The following operations can be performed to set access control policies for the two networks by configuring the following objects.

1. Check ENUM server information. For details, see Section 4.1.1 on page 68.
2. Create *EnumZone* objects.

```
IPWorks> create enumzone 1 -set enumzonename="5.4.e164.arpa"
IPWorks> create enumzone 2 -set enumzonename="6.4.e164.arpa"
IPWorks> create enumzone 3 -set enumzonename="6.8.e164.arpa"
```

3. Create *EnumAcl* objects.

```
IPWorks> create enumacl 1 -set aclname=acl1;matchlist="{10.0.0.0/16;}"
IPWorks> create enumacl 2 -set aclname=acl2;matchlist="{12.0.0.0/16;}"
```

4. Create *EnumView* objects.

```
IPWorks> create enumview 1 -set viewname=internalview;rank=1
00;aclid=1;serveridlist="1"
IPWorks> create enumview 2 -set viewname=externalview;rank=20
0;aclid=2;serveridlist="1,2"
```

5. Create *EnumZVRel* objects to link ENUM Zones and ENUM Views together.

```
IPWorks> create enumzvrel -set zoneid=1;viewid=1
IPWorks> create enumzvrel -set zoneid=2;viewid=1
IPWorks> create enumzvrel -set zoneid=2;viewid=2
```

Note: Although it is possible to have an overlapping configuration allowing many ENUM Views to control the same ENUM Zone, it is not recommended because it slows down the ENUM query performance.



The configuration above enables the internal clients to query *EnumDnSched* both in ENUM Zone 1 and 2 (using *internalview*), and blocks ENUM Zone 1 from the external clients, so that only ENUM Zone 2 is available (using *externalview*). Since ENUM Zone 3 is under control of the default view, the *EnumDnSched* is accessible by any client.

4.3 Configuring ENUM Query Preference

In response to an ENUM query, ENUM server searches for *EnumDnSched* according to the query content. If the corresponding *EnumDnSched* is not found, ENUM server searches for the *EnumDn* scope, and the query answer is the best matched *EnumDn* scope. If the *EnumDn* scope is not found, ENUM server searches for ENUM wildcard, and the query answer is the best matched ENUM wildcard.

EnumDn scope specifies a number range for the *EnumDnRange* object. For example, an *EnumDnRange* *3.2.1.e164.ip.telco.com* is in the *EnumZone* *e164.ip.telco.com* and the scope is set to *3241~5233*. It indicates that the object provides common information for all numbers from *1233241* to *1235233*.

ENUM wildcard uses a wildcard expression to represent a number series. For example, **.8.e164.ip.telco.com* is a wildcard expression, which stands for different telephone numbers with the same ending of *8.e164.ip.telco.com*, such as *8.e164.ip.telco.com*, *7.8.e164.ip.telco.com*, *8.8.e164.ip.telco.com*, *6.7.8.e164.ip.telco.com*, and so on.

The following example shows how ENUM query preference works.

Suppose there is a sample network containing an ENUM server. The following operations give the query results received under different configurations. To run this example, Storage Server and ENUM server must be in the running status.

1. Check ENUM server information. For details, see Section 4.1.1 on page 68.
2. Create an *EnumZone* object.

```
IPWorks> create enumzone 1 -set enumzonename
=8.e164.ip.telco.com
```

3. Create the first ENUM wildcard **.7.8.e164.ip.telco.com*.

```
IPWorks> create enumdnrange 7.8.e164.ip.telco.com -set naptrorder=10;
naptrpreference=20;naptrflags=nU;naptrservice=E2U+SIP;naptrTxt
=!^.*$!sip:87230000@ip.telco.com!
```

4. Query the telephone number *1.2.3.4.5.6.7.8.e164.ip.telco.com*.



```
# dig @127.0.0.1 1.2.3.4.5.6.7.8.e164.ipitelco.com NAPTR
; <<>> DiG 9.9.8-P3 <<>> @127.0.0.1 1.2.3.4.5.6.7.8.e164.
ipitelco.com NAPTR
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 1033
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. IN NAPTR

;; ANSWER SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. 0 IN NAPTR 10 20 "U"
"E2U+SIP" "!^.*$!sip:87230000@ipitelco.com!" .

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Oct 23 10:27:45 2008
;; MSG SIZE rcvd: 109
```

5. Create the second ENUM wildcard *.5.6.7.8.e164.ipitelco.com.

```
IPWorks> create enumdnrange 5.6.7.8.e164.ipitelco.com -set naptrorder=20;
naptrpreference=30;naptrflags=nU;naptrservice=E2U+SIP;naptrTxt
=!^.*$!sip:87651111@ipitelco.com!
```

6. Query the telephone number 1.2.3.4.5.6.7.8.e164.ipitelco.com.

```
# dig @127.0.0.1 1.2.3.4.5.6.7.8.e164.ipitelco.com NAPTR
; <<>> DiG 9.9.8-P3 <<>> @127.0.0.1 1.2.3.4.5.6.7.8.e164.
ipitelco.com NAPTR
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 1294
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. IN NAPTR

;; ANSWER SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. 0 IN NAPTR 20 30 "U"
"E2U+SIP" "!^.*$!sip:87651111@ipitelco.com!" .

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Oct 22 09:44:33 2008
;; MSG SIZE rcvd: 109
```

7. Create the first EnumDn scope.

```
IPWorks> create enumdnrange 5.6.7.8.e164.ipitelco.com -set
scope=4321~9321;naptrorder=20;
naptrpreference=30;naptrflags=nU;naptrservice=E2U+SIP;naptrTxt
=!^.*$!sip:87651112@ipitelco.com!
```

8. Query the telephone number 1.2.3.4.5.6.7.8.e164.ipitelco.com.



```
# dig @127.0.0.1 1.2.3.4.5.6.7.8.e164.ipitelco.com NAPTR
; <<>> DiG 9.9.8-P3<<>> @127.0.0.1 1.2.3.4.5.6.7.8.e164.
ipitelco.com NAPTR
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 1294
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;1.2.3.4.5.6.7.8.e164.ipitelco.com. IN NAPTR

;; ANSWER SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. 0 IN NAPTR 20 30 "U"
"E2U+SIP" "!^.*$!sip:87651112@ipitelco.com!" .

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Oct 22 09:44:33 2008
;; MSG SIZE rcvd: 109
```

9. Create the second EnumDn scope.

```
IPWorks> create enumdnrange 5.6.7.8.e164.ipitelco.com -set
scope=4321-5321;naptrorder=20;
naptrpreference=30;naptrflags=nU;naptrservice=E2U+SIP;naptrTxt
=!^.*$!sip:87651113@ipitelco.com!
```

10. Query the telephone number 1.2.3.4.5.6.7.8.e164.ipitelco.com.

```
# dig @127.0.0.1 1.2.3.4.5.6.7.8.e164.ipitelco.com NAPTR
; <<>> DiG 9.9.8-P3 <<>> @127.0.0.1 1.2.3.4.5.6.7.8.e164.
ipitelco.com NAPTR
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 1294
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;1.2.3.4.5.6.7.8.e164.ipitelco.com. IN NAPTR

;; ANSWER SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. 0 IN NAPTR 20 30 "U"
"E2U+SIP" "!^.*$!sip:87651113@ipitelco.com!" .

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Oct 22 09:44:33 2008
;; MSG SIZE rcvd: 109
```

11. Create an EnumDnSched object.

```
IPWorks> create enumdnsched 1.2.3.4.5.6.7.8.e164.ipitelco.com
-set naptrorder=30;
naptrpreference=40;naptrflags=nU;naptrservice=E2U+SIP;naptrTxt
=!^.*$!sip:87654321@ipitelco.com!
```

12. Query the telephone number 1.2.3.4.5.6.7.8.e164.ipitelco.com.



```
# dig @127.0.0.1 1.2.3.4.5.6.7.8.e164.ipitelco.com NAPTR
; <<>> DiG 9.9.8-P3 <<>> @127.0.0.1 1.2.3.4.5.6.7.8.e164.
ipitelco.com NAPTR
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 214
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 0

;; QUESTION SECTION:
;1.2.3.4.5.6.7.8.e164.ipitelco.com. IN NAPTR

;; ANSWER SECTION:
1.2.3.4.5.6.7.8.e164.ipitelco.com. 0 IN NAPTR 30 40 "U"
"E2U+SIP" "!^.*$!sip:87654321@ipitelco.com!" .

;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Oct 22 09:42:37 2008
;; MSG SIZE rcvd: 109
```

The above example shows the following query preference (<High Preference> > <Low Preference>):

```
1.2.3.4.5.6.7.8.e164.ipitelco.com > 5.6.7.8.e164.ipitelco.com
(scope 4321~5321) > 5.6.7.8.e164.ipitelco.com (scope 4321~9321) >
*.5.6.7.8.e164.ipitelco.com > *.7.8.e164.ipitelco.com
```

4.4 Configuring Number Portability

Number Portability (NP) is a telecommunication network feature that enables end users to retain their telephone numbers when changing service providers, service types, or locations.

In IPWorks NP solution, ENUM supports: forwarding a query to an external SS7/LDAP database, such as Service Control Point (SCP), according to the relevant specifications; receiving the reply from the database; and responding to the client. Figure 4 shows an ERH module, which is bound to the ENUM server, communicates with the external SS7/LDAP database.

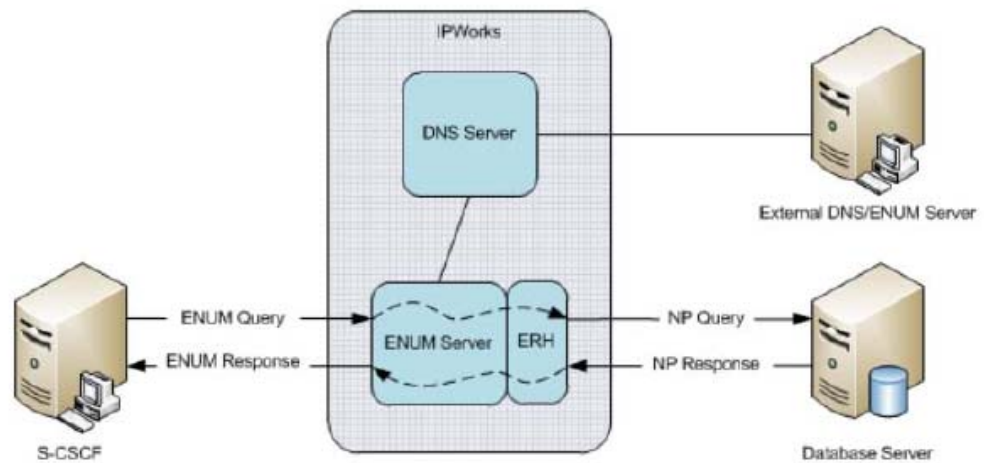


Figure 4 ERH Residing in ENUM Server

The following steps show that how ENUM NP works after the ENUM server receives a query:

1. ENUM server finds the best matched zone by zone name from the ENUM NP/Toll Free Request.
2. ENUM server checks the ENUM configuration to determine if the number series will be forwarded to the external SS7 database or external LDAP database.
3. ENUM server prepares data for the NP Query.
4. ENUM server sends NP/Toll Free query to ERH Handler, and then ERH interrogates external SS7/LDAP database to see if this number has been ported out.
5. ENUM server communicates with ERH Module, receives the reply from ERH Module and formats NAPTRRecord according to RFC4694 and RFC4769, and then sends the ENUM response back to Client.

Currently, ERH provides NP functions based on the following protocols:

- AIN
- MAP
- INAP
- LDAP

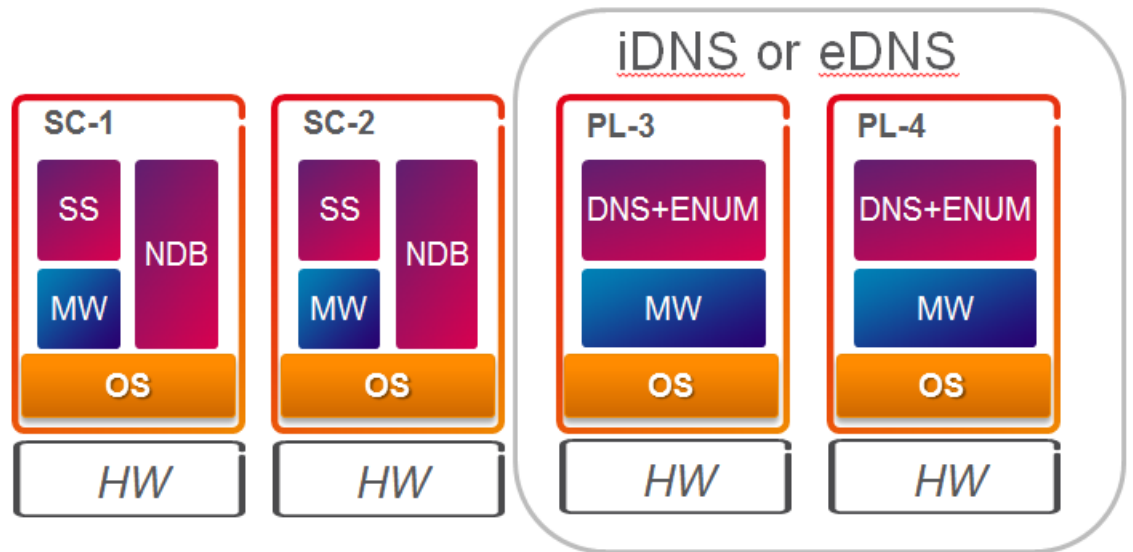


Figure 5 Reference Configurations

The configuration of NP contains the following topics:

- Global Configuration for NP Protocol
- Configuring NP Using AIN Protocol
- Configuring NP Using MAP Protocol
- Configuring NP Using INAP Protocol
- Configuring NP Using LDAP Protocol

4.4.1 Global Configuration for NP Protocol

This section includes the following topics:

- Setting Country Code for NP
- ERH Failure Discarding for NP

4.4.1.1 Setting Country Code for NP

The country code is a global setting for NP function, and it is valid for the protocols: AIN, MAP, INAP, and LDAP.

```
IPWorks> create countrycode -set ccode=86
1 object(s) created.
```



4.4.1.2 ERH Failure Discarding for NP

The ERH Failure Discarding is a global setting for NP function, and it is valid for the protocols: AIN, MAP, INAP, and LDAP.

When the local or remote SS7 stack or LDAP database has problem or stopped, ENUM server discards the ERH failure and the ENUM client does not receive any response from ENUM server.

To keep the ERH failure with SS7 stack:

1. Log on to the ECLI:

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `discardErhFailure` of the MO *Erh*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
(config-Erh=1)>discardErhFailure=false
```

4. Commit the change.

```
(config-Erh=1)>commit
```

Note: The change of the attribute takes effect after the ENUM server is restarted.

To keep the ERH failure with LDAP database:

Result: The ENUM server keeps the ERH failures and returns NXDOMAIN to ENUM client when ERH failed.

4.4.2 Configuring NP Using AIN Protocol

To enable NP function using the AIN protocol in ENUM server, the *AINNode* object needs to be configured. *AINNode* holds server level configuration of NP using AIN protocol.

1. Check whether *ServerType* of ENUM server is configured properly.



```
ssh <username>@<MIP_OAM_IP> -t -s cli
>show -v ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,ServerType=1
ServerType=1
  Server1 <default>
    "PL-3"
    "PL-4"
  Server2 <default>
    ""
    ""
  serverTypeId="1" <default>
```

2. Create an *AINNode* object.

```
IPWorks> create AINNode -set EnumServerId=1;AINUserId=
1234567890;LocalSSN=10;
LocalSPC=100;BearerCapability=0;NPSSwitch=1
1 object(s) created.
```

Note: AINUserId is the directory number in the signaling network that represents the NP SSP node.

- LocalSSN and LocalSPC must be configured the same as those in the local SS7 Stack.
- BearerCapability must be set according to the remote SCP configuration.
- The NP function using the AIN protocol is enabled when NPSSwitch=1. If NPSSwitch=0, the function is disabled.

To configure LNP feature and TollFree feature for AIN, follow the following topics:

- Configuring the LNP Feature
- Configuring the TollFree Feature

To configure SCCP Classes for AIN, follow the following topics:

- Configuring SCCP Classes for AIN

4.4.2.1 Configuring the LNP Feature

To handle the NP queries and construct the response to ENUM client, the server level needs to be configured. Since an ENUM number series is held in a range of ENUM numbers, the *EnumZone*, *EnumDNRange* and *DestNode* objects are required in this configuration.

For AIN configuration, LNP feature have two types of response format, 'TEL' and 'SIP':

- Configuring NP Using AIN for LNP with 'TEL' URI Format in Response
- Configuring NP Using AIN for LNP with 'SIP' URI Format in Response



4.4.2.1.1 Configuring NP Using AIN for LNP with 'TEL' URI Format in Response

1. Create an *EnumZone* object.

```
IPWorks> create enumzone -set enumzoneid=1;enumzonename=e164.arpa
1 object(s) created.
```

2. Create an *AINLNpData* object and set 'TEL' URI format.

```
IPWorks> create AINLNpData -set ProtocolSettingName=ainlnpdata1;ServiceParameter=E2U+pstn:tel
1 object(s) created.
```

3. Create a *DestNode* object and associate it to the *AINLNpData* created in Step 2.

```
IPWorks> create destnode -set destnodename=lnpdestnode1;type=7;protocolsettingname=ainlnpdata1
1 object(s) created.
```

4. Create an *EnumDNRange* object in the *EnumZone* created in Step 1 and associate it to the *DestNode* created in Step 3.

```
IPWorks> create enumdnrange 1.2.3.4.e164.arpa -set destnode=lnpdestnode1
1 object(s) created.
```

4.4.2.1.2 Configuring NP Using AIN for LNP with 'SIP' URI Format in Response

The configuration for LNP with 'SIP' URI format is different from that with 'TEL' URI format. In the configuration for LNP with 'SIP' URI format, 'SIP' URI format and domain name need to be set at the same time when creating the *AINLNpData* object. Other steps are the same as those in the configuration for LNP with 'TEL' URI format.

```
IPWorks> create AINLNpData -set ProtocolSettingName=ainlnpdata1;ServiceParameter=E2U+pstn:sip;Domain=test.com
1 object(s) created.
```

4.4.2.2 Configuring the TollFree Feature

The server level configuration for LNP is *AINTollFreeData* object. The *EnumZone*, *EnumDNRange* and *DestNode* objects are also needed.

For AIN configuration, TollFree feature have two types of response format, 'TEL' and 'SIP':

- Configuring NP Using AIN for TollFree with 'TEL' URI Format in Response
- Configuring NP Using AIN for TollFree with 'SIP' URI Format in Response



4.4.2.2.1 Configuring NP Using AIN for TollFree with 'TEL' URI Format in Response

1. Create an *EnumZone* object.

```
IPWorks> create enumzone -set enumzoneid=2;enumzonename=tollfree.e164.arpa
1 object(s) created.
```

2. Create an *AINTollFreeData* object and set 'TEL' URI format.

```
IPWorks> create aintollfreedata -set
protocolsettingname=aintollfreedata1;carrier=cs:1;nc:0;dig:;
chargenumber=nn:3;np:1;pres:0;scr:0;dig:123;
chargepartystationtype=0;lata=nn:3;np:1;scr:0;dig:456;
translationtype=11;ServiceParameter=E2U+pstn:tel
1 object(s) created.
```

3. Create a *DestNode* and associate it to the *AINTollFreeData* created in Step 2.

```
IPWorks> create destnode -set destnodename=tollfreedestnode1;type=8;
protocolsettingname=aintollfreedata1
1 object(s) created.
```

4. Create an *EnumDNRange* object in the *EnumZone* created in Step 1 and associate it to the *DestNode* created in Step 3.

```
IPWorks> create enumdnrange 1.2.3.4.tollfree.e164.arpa -set
destnode=tollfreedestnode1
1 object(s) created.
```

4.4.2.2.2 Configuring NP Using AIN for TollFree with 'SIP' URI Format in Response

The AIN configuration for TollFree with 'SIP' URI format in response must be compatible with RFC 4694 and RFC 4769.

The configuration for TollFree with 'SIP' URI format is different from that with 'TEL' URI format. In the configuration for TollFree with 'SIP' URI format, 'SIP' URI format and domain name need to be set at the same time when creating the *AINTollFreeData* object. The other steps are the same as those in the configuration for TollFree with 'TEL' URI format.

- RFC 4694 and RFC 4769 compliant:

```
IPWorks> create aintollfreedata -set
protocolsettingname=aintollfreedata1;carrier=cs:1;nc:0;dig:;
chargenumber=nn:3;np:1;pres:0;scr:0;dig:123;chargepartystationtype=0;
lata=nn:3;np:1;scr:0;dig:456;translationtype=11;
ServiceParameter=E2U+pstn:sip;Domain=test.com
1 object(s) created.
```



4.4.2.3 Configuring SCCP Classes for AIN

To configure the acceptable quality of the service provided by the underlying layers of IPWorks, the attribute `ainQualityOfService` under the MO `ErhSs7` needs to be set as following:

1. Log on to the ECLI:

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `ainQualityOfService` of the MO `ErhSs7`.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
,ErhSs7=1
```

```
(config-ErhSs7=1)> ainQualityOfService=<QoS>
```

Note: The value `QoS` is an integer from 0 to 3.

4. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change of the attribute takes effect after the ENUM server is restarted.

4.4.3 Configuring NP Using MAP Protocol

To enable NP function using MAP protocol in ENUM server, the `MAPNode` object is required. The object holds server level configuration of NP using MAP protocol.

1. Check whether `ServerType` of ENUM server is configured properly.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
>show -v ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,ServerType=1
ServerType=1
  Server1 <default>
    "PL-3"
    "PL-4"
  Server2 <default>
    ""
    ""
  serverTypeId="1" <default>
```

2. Create an `MAPNode` object.



```
IPWorks> create MAPNode -set EnumServerId=1;GsmScfAddre  
ss=tn:1:np:1:dig:9ab*6;  
LocalSSN=10;LocalSPC=100;InternationalFormat=0;NPSTch=1  
1 object(s) created.
```

'TEL' format and 'SIP' format are used for the server level configuration for the NP function using MAP protocol. To configure the format, follow the following topics:

- Configuring NP Using MAP with 'TEL' URI Format in Response
- Configuring NP Using MAP with 'SIP' URI Format in Response

For further MAP configuration, follow the following topics:

- Configuring NP Using MAP with Customized Digit Table
- Configuring NP Using MAP with Remote SSN for Called Party Address
- MAP Configuration for Number Format and MSISDN Format
- MAP Configuration for Sending SRI Message Instead of ATI Message
- MAP Configuration for Interrogation Type in SRI Message
- MSRN Handling Configuration in SRI Response
- Configuring routing number format in ENUM response for MAP

4.4.3.1 Configuring NP Using MAP with 'TEL' URI Format in Response

The server level configuration for MAP is *MAPData* object. The *EnumZone*, *EnumDNRange* and *DestNode* objects are also needed.

1. Create an *EnumZone* object.

```
IPWorks> create enumzone -set enumzoneid=3;enumzonename=map.e164.arpa  
1 object(s) created.
```

2. Create a *MAPData* object and set 'TEL' URI format.

```
IPWorks> create MAPData -set ProtocolSettingName=mapdata1;Translat  
ionType=45;ServiceParameter=E2U+pstn:tel  
1 object(s) created.
```

3. Create a *DestNode* and associate it to the *MAPData* created in Step 2.

```
IPWorks> create destnode -set destnodename=mapdestnode1;type=  
9;protocolsettingname=mapdata1  
1 object(s) created.
```

4. Create an *EnumDNRange* object in the *EnumZone* created in Step 1 and associate it to the *DestNode* created in Step 3.



```
IPWorks> create enumdnrange 1.2.3.4.map.e164.arpa -set destnode=mapdestnode1
1 object(s) created.
```

4.4.3.2 Configuring NP Using MAP with 'SIP' URI Format in Response

Two formats in the configuration for MAP with 'SIP' URI in response are used according to the value of the attribute `replyformat` of the `MAPData` object.

The configuration for MAP with 'SIP' URI format is different from that with 'TEL' URI format. In the configuration for MAP with 'SIP' URI format, 'SIP' URI format and `replyformat` need to be set, and `subdomain` or `sipNpDomain` needs to be specified according to the value of `replyformat` at the same time. Other steps are the same as those in the configuration for MAP with 'TEL' URI format.

- Using format one in response

```
IPWorks> create MAPData -set
ProtocolSettingName=mapdata1;TranslationType=45;
ServiceParameter=E2U+pstn:sip;replyformat=1;sipnpdomain=test.com
1 object(s) created.
```

- Using format two in response

```
IPWorks> create MAPData -set ProtocolSettingName=mapda
tal;TranslationType=45;
ServiceParameter=E2U+pstn:sip;replyformat=2;subdomain=test.com
1 object(s) created.
```

4.4.3.3 Configuring NP Using MAP with Customized Digit Table

To configure the MAP with Customized Digit Table:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attributes `enableCustomizedDigTable` and `mapDigitTable` of the MO `ErhSs7`.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1,ErhSs7=1
(config-ErhSs7=1)>enableCustomizedDigTable=true
(config-ErhSs7=1)>mapDigitTable="0x0=0,0x1=1,0x2=2,0x3=3,0x4=4,0x5=5,0
x6=6,0x7=7,0x8=8,0x9=9,0xA=a,0xB=b,0xC=c,0xD=d ,0xE=e"
```

4. Commit the changes.

```
(config-ErhSs7=1)>commit
```

Note: These changes will take effect after the ENUM server is restarted.



4.4.3.4 Configuring NP Using MAP with Remote SSN for Called Party Address

This section provides an example of updating the `remoteSSN` attribute by using ECLI.

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `remoteSSN` of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,  
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1,ErhSs7=1  
(config-ErhSs7=1)>remoteSSN=10
```

4. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.

4.4.3.5 MAP Configuration for Number Format and MSISDN Format

The default routing number format and MSISDN format in ENUM response is like "[countrycode]-[sec1]-[secN]", such as "+86-234-567-8901". However, some customers prefer the routing number format or MSISDN format without hyphen, such as "+862345678901". To meet the requirement, do the following:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `MAPResponstNumberFormat` of the MO *Erh*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1  
, IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
```

```
(config-Erh=1)>MAPResponstNumberFormat=CALLEDPARTYNUM  
BER
```

4. Commit the change.

```
(config-Erh=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.



4.4.3.6 MAP Configuration for Sending SRI Message Instead of ATI Message

By default, ATI message will be sent. However, some customer has NPDB that only supports Sending Routing Information (SRI) MAP operation. To meet the requirement, do the following:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute MAPMessage of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=
1, IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh
=1,ErhSs7=1
```

```
(config-ErhSs7=1)>MAPMessage=MAP_SRI
```

4. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.

4.4.3.7 MAP Configuration for Interrogation Type in SRI Message

By default, basic interrogation is used. To use forwarding interrogation, do the following:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute InterrogationType of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=
1, IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh
=1,ErhSs7=1
```

```
(config-ErhSs7=1)>InterrogationType=FORWARDING
```

4. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.



4.4.3.8 MSRN Handling Configuration in SRI Response

The routing number may be contained in Mobile Station Roaming Number (MSRN) element in the SRI response from NPDB. By default, the MSRN is sent to ENUM without any changes as routing number. Besides, the routing number can be extracted from MSRN and then sent to ENUM. Set the length of routing number to extract the routing number from MSRN. The length is configurable and set to 4 by default.

To extract the routing number from MSRN, do the following:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration.

```
>configure
```

3. Change the attribute MSRNMode of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=
1, IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh
=1,ErhSs7=1
```

```
(config-ErhSs7=1)>MSRNMode=RNHANDLING
```

4. Change the attribute RNLength of the MO *ErhSs7* if needed, the default value is 4.

```
(config-ErhSs7=1)>RNLength=4
```

5. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.

4.4.3.9 Configuring Routing Number Format in ENUM Response for MAP

This section provides an example of updating the RNwithCountryCode attribute by using ECLI.

1. Log on to the ECLI

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute remoteSSN of the MO *ErhSs7*.



```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1
, IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
```

```
(config-Erh=1)>RNwithCountryCode=true
```

4. Commit the change.

```
(config-Erh=1)>commit
```

Note: The change will take effect after the ENUM server is restarted.

4.4.4 Configuring NP Using INAP Protocol

To enable NP function using INAP protocol in ENUM server, the *INAPNode* object is required. The object holds server level configuration of NP using INAP protocol.

1. Check whether *ServerType* of ENUM server is configured properly.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
>show -v ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,ServerType=1
ServerType=1
  Server1 <default>
    "PL-3"
    "PL-4"
  Server2 <default>
    ""
    ""
  serverTypeId="1" <default>
```

This sample example indicates that the EnumServer 1 is on PL-3 and PL-4.

2. Create an *INAPNode* object .

```
IPWorks> create INAPNode -set EnumServerId=1;ServiceKey=32456;LocalSSN=10;
LocalSPC=100;InternationalFormat=0;NPSSwitch=1
1 object(s) created.
```

'TEL' format and 'SIP' format are used for the server level configuration for the NP function using INAP protocol. To configure the format, follow the following topics:

- Configuring NP Using INAP with 'TEL' URI Format in Response
- Configuring NP Using INAP with 'SIP' URI Format in Response

For further INAP configuration, follow the following topics:

- Configuring NP Using INAP with Customized Digit Table
- Configuring NP Using INAP for Routing Number Format
- Configuring NP Using INAP for Application Context



4.4.4.1 Configuring NP Using INAP with 'TEL' URI Format in Response

INAPData object is used to configure the number server level data for an INAP query. The *EnumZone*, *EnumDNRange* and *DestNode* objects are also needed.

1. Create an *EnumZone* object.

```
IPWorks> create enumzone -set enumzoneid=4;enumzonename=inap.e164.arpa
1 object(s) created.
```

2. Create an *SCCPAddress* object.

```
IPWorks> create sccpaddress cdpa -set GlobalTitleIndicator=2;translationtype=123
1 object(s) created.
```

3. Create an *INAPData* object and set 'TEL' URI format.

```
IPWorks> create INAPData -set ProtocolSettingName=inapdata1;ServiceParameter=E2U+pstn:tel;addressname=cdpa
1 object(s) created.
```

4. Create a *DestNode* object and associate it to the *INAPData* created in Step 3.

```
IPWorks> create destnode -set destnodename=inapdestnode1;type=10;protocolsettingname=inapdata1
1 object(s) created.
```

5. Create an *EnumDNRange* object in the *EnumZone* created in Step 1 and associate it to the *DestNode* created in Step 4.

```
IPWorks> create enumdnrange 1.2.3.4.inap.e164.arpa -set destnode=inapdestnode1
1 object(s) created.
```

4.4.4.2 Configuring NP Using INAP with 'SIP' URI Format in Response

The configuration for INAP with 'SIP' URI format is different from that with 'TEL' URI format. 'SIP' URI format and domain name need to be set when creating the *INAPData* object. Other steps are the same as those in the configuration for INAP with 'TEL' URI format.

```
IPWorks> create INAPData -set ProtocolSettingName=inapdata1;ServiceParameter=E2U+pstn:sip;Domain=test.com;addressname=cdpa
1 object(s) created.
```

4.4.4.3 Configuring NP Using INAP with Customized Digit Table

To configure the INAP with Customized Digit Table:

1. Log on to the ECLI.



```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attributes `enableCustomizedDigTable` and `inapDigitTable` of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1,ErhSs7=1
(config-ErhSs7=1)>enableCustomizedDigTable=true
(config-ErhSs7=1)>inapDigitTable="0x0=0,0x1=1,0x2=2,0x3=3,0x4=4,0x5=5,
0x6=6,0x7=7,0x8=8,0x9=9,0xA=a,0xB=b,0xC=c,0xD=d,0xE=e"
```

4. Commit the changes.

```
(config-ErhSs7=1)>commit
```

Note: The changes of the attributes take effect after the ENUM server is restarted.

4.4.4.4

Configuring NP Using INAP for Routing Number Format

The default routing number format in ENUM response is like "+[countrycode]-[sec1]-[secN]", such as "+86-234-567-8901". However, for some customers who prefer the routing number format as "+[Called party number]", such as "+862345678901" where no hyphen is used. To use the non-default routing number format, change `formatFlag` by using ECLI:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `formatFlag` of the MO *ErhSs7* to true.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1,ErhSs7=1
(config-ErhSs7=1)>formatFlag=true
```

4. Commit the change.

```
(config)>commit
```

Note: The change of the attribute takes effect after the ENUM server is restarted.



4.4.4.5 Configuring NP Using INAP for Application Context

The default Application Context that INAP uses is from CS1. To change source from CS1 to CS1+, perform the following steps:

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enter the configuration mode.

```
>configure
```

3. Change the attribute `inapApplicationContextType` of the MO *ErhSs7*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1,ErhSs7=1
(config-ErhSs7=1)>inapApplicationContextType=CS1PLUS
```

4. Commit the change.

```
(config-ErhSs7=1)>commit
```

Note: The change of the attribute takes effect after the ENUM server is restarted.

4.4.5 Configuring NP Using LDAP Protocol

To enable NP function using LDAP protocol in ENUM server:

1. Check whether *ServerType* of ENUM server is configured properly.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
>show -v ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,ServerType=1
ServerType=1
  Server1 <default>
    "PL-3"
    "PL-4"
  Server2 <default>
    ""
    ""
  serverTypeId="1" <default>
```

This sample example indicates that the `EnumServer 1` is on PL-3 and PL-4.

2. Log on to the ECLI interface.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

For information about how to use ECLI, refer to *Ericsson Command-Line Interface User Guide*.

3. Enable the LDAP by using ECLI.



```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
(config-Erh=1)>ldap=true
(config-Erh=1)>commit
```

4. Configure the LDAP Dictionary by following command:

```
#cat /etc/ipworks/ldapschema/ldap_dictionary.xml
```

```
...
<service name="NP">
<cudbRootEntry name="dc=example,dc=com"/>
<searchDn name="MSISDN=%s,ou=people,"/>
<entryList>
<entry name="NphData">
<dn name="MSISDN=%s,ou=people,"/>
<attr name="NPREFIX" alias="NPREFIX"/>
<attr name="SUBSTYPE" alias="SUBSTYPE"/>
</entry>
</entryList>
<searchObject name="NphData">
<searchRequest baseDn="searchDn" searchScope="0" filter="objectC
<searchResult>
<entry ref="NphData"/>
</searchResult>
</searchObject>
</service>
...
```

Make sure that the content is aligned with customer LDAP server configuration: `cudbRootEntry`, `searchDn` and `dn`.

'TEL' format and 'SIP' format are used for the server level configuration for the NP function using LDAP protocol. To configure the format, follow the following topics:

- Configuring NP Using LDAP with 'TEL' URI Format in Response
- Configuring NP Using LDAP with 'SIP' URI Format in Response

4.4.5.1 Configuring NP Using LDAP with 'TEL' URI Format in Response

1. Create an *EnumZone* object.

```
# ipwcli
IPWorks> create enumzone -set
enumzoneid=4;enumzonename=ldap.e164.arpa
1 object(s) created.
```

2. Create an *EnumDNRange* object in the EnumZone created in the previous step, and set *DestNode* to `ldap`.



```
IPWorks> create enumdnrange 1.2.3.4.ldap.e164
.arpa -set destnode=ldap
1 object(s) created.
```

3. Exit the IPWorks CLI.

```
IPWorks> exit
```

4. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

5. Configure the attribute *serviceParameter* of the MO *ErhLdap*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,
Erh=1,ErhLdap=1
(config-ErhLdap=1)>serviceParameter="E2U+pstn:tel"
(config-ErhLdap=1)>commit
```

6. Configure CUDB connection pool with NP. For details about how to configure CUDB connection pool, see Section 4.7.8 Configuring CUDB Connection Pool on page 124.

Note: Since this configuration is specified for NP, then make sure the *<Service Name>* of *CudbServiceSite* is NP during the configuration.

4.4.5.2 Configuring NP Using LDAP with 'SIP' URI Format in Response

Two formats in the configuration for LDAP with 'SIP' URI in response are used according to the attribute *replyformat* of the MO *ErhLdap*.

The configuration for LDAP with 'SIP' URI format is different from the configuration with 'TEL' URI format. In the configuration for LDAP with 'SIP' URI format, 'SIP' URI format and *replyformat* must be set, and *subdomain* or *sipNpDomain* must be specified according to the value of *replyformat* at the same time. Other steps are as same as those in the configuration for LDAP with 'TEL' URI format.

For detailed description of relations among *ServiceParameter*, *replyformat*, *subdomain* and *sipNpDomain*, refer to the *ErhLdap* in *Managed Object Model (MOM)*.

Using Format One in Response

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Configure the MO *ErhLdap*.



```
>configure
(config)> dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,
Erh=1,ErhLdap=1
(config-ErhLdap=1)> serviceParameter="E2U+pstn:sip"
(config-ErhLdap=1)> replyFormat=1
(config-ErhLdap=1)> sipNpDomain="test.com"
(config-ErhLdap=1)> commit
```

Using Format Two in Response

1. Log on to the ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Configure the MO *ErhLdap* by using ECLI.

```
>configure
(config)> dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,
Erh=1,ErhLdap=1
(config-ErhLdap=1)>serviceParameter="E2U+pstn:sip"
(config-ErhLdap=1)>replyFormat=2
(config-ErhLdap=1)>subDomain="test.com"
(config-ErhLdap=1)>commit
```

4.5 Configuring RCSe Interconnect

This section describes how to configure the RCSe Interconnect functionality.

Note: Currently, the NP configuration for the RCSe Interconnect functionality is only based on the MAP protocol.

To configure the NP for the RCSe Interconnect functionality, perform the following steps:

1. Enable the RCSe Interconnect functionality by using ECLI.

- a. Log on to ECLI.

```
ssh <username>@<MIP_OAM_IP> -t -s cli
```

- b. Enter the configuration mode.

```
>configure
```

- c. Change the attribute *rcseInterConnect* of the MO *Erh*.

```
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1,Erh=1
(config-Erh=1)>rcseInterConnect=true
```

- d. Commit the change.



```
(config-Erh=1)>commit
```

Note: The change of the attribute takes effect after the ENUM server is restarted.

2. Create the *MAPNode* and other ENUM-related objects to configure the traditional NP function by using MAP protocol.

For details, see Section 4.4.3 on page 101.

3. Create the *EnumOperator* object and configure the *EnumClientRealm* object.

For details, see Section 4.1.20.1 on page 84 and Section 4.1.21 on page 87.

Note: When the users are creating the *EnumOperator* object and configuring the *EnumClientRealm* object, need to pay attention that the NP response format might vary based on various combinations of the terminating operator type and the client type. Table 3 lists the combination rules.

Table 3 NP Response Consisting of Specific URI Format

| Type of Terminating Operator | Type of ENUM Client | |
|-------------------------------------|---|---|
| | Local or National ENUM Clients | International ENUM Clients or Unknown Clients |
| Local Operators (OP) or unknown OPs | The expected NP response example: E2U+pstn:tel !^.*\$!tel:+86-123-456-7809;npdi;rn=+86-234-567-8901!!, where the E2U+PSTN:tel is a fixed URI format. | NXDOMAIN |



Table 3 NP Response Consisting of Specific URI Format

| | | |
|--|--|---|
| Other OPs not participating in RCSe interconnect | <p>Two selective NP response:</p> <ul style="list-style-type: none"> • If the users apply the attribute <code>ServiceParameter</code> with <code>E2U+PSTN:tel</code>, the expected NP response example is: <code>E2U+pstn:tel !^.*\$!tel:+86-123-456-7809;npdi;rn=+86-234-567-8901!!</code> • If the users apply the attribute <code>ServiceParameter</code> with <code>E2U+PSTN:sip</code>, the expected NP response example is: <code>E2U+pstn:sip!^.*\$!sip:+86-123-456-7809;npdi;rn=+86-234-567-8901@ims.mnc004.mcc460.3gppnetwork.org;user=phone!.</code>⁽¹⁾ | NXDOMAIN |
| Other OPs participating in RCSe interconnect | The expected NP response is generated by server itself, which means the users cannot make any additional configuration. The NP response contains a fixed URI <code>E2U+SIP</code> . | <ul style="list-style-type: none"> • For International ENUM Clients: E2U + SIP • For unknow clients: NXDOMAIN |

(1) In the NP response, the `ims` is the Subdomain value that takes effective when `ServiceParameter` is with value `E2U+PSTN:sip`.

4.6 Configuring Split Namespace

Split Namespace in DNS/ENUM is a mechanism to ensure network security. Under this setting, records are allowed to configure differently according to the namespace. For example, a NAPTR record can be defined differently in different views in ISC BIND Server.

However, ENUM server does not have such a mechanism to specify namespaces for ENUM objects. Thus a security pitfall can be found under the following configuration:

One ENUM server handles both internal and external queries. When an NP query is coming, ENUM server uses its ERH handler to forward the query to the external database SCP via SS7 protocol. Because SCP is often located in the external network, it is not safe to allow the direct communication between SCP and the machine on which ENUM server and ERH handler are installed.

To solve this security problem, the two DNS servers are deployed in the internal and external network respectively. Figure 6 shows the locations of the two servers in the split namespace architecture. The modules of the two servers are shown in Table 4. With this deployment, the DNS server in IPWorks1 (internal) is protected from the external network because there is no direct communication between SCP and the DNS server.

This deployment can be applied to different scenarios. The most important scenario is for NP query. If an NP query comes from the external network, the ENUM server in IPWorks2 (external) forwards the query to SCP via ERH. For details on the NP query handling, see Section 4.4 on page 94. If an NP query comes from the internal network, the ENUM server in IPWorks1 (internal) forwards the query to the DNS Server in IPWorks1 (internal), DNS Server in IPWorks1 (internal) forwards the query to the ENUM server in IPWorks2 (external), and ENUM server in IPWorks2 (external) forwards the query to the ERH Server in IPWorks2 (external). Then, the query is handled in the same way as an external query except that the response returns to the DNS in IPWorks1 (internal) via the ENUM in IPWorks2 (external).

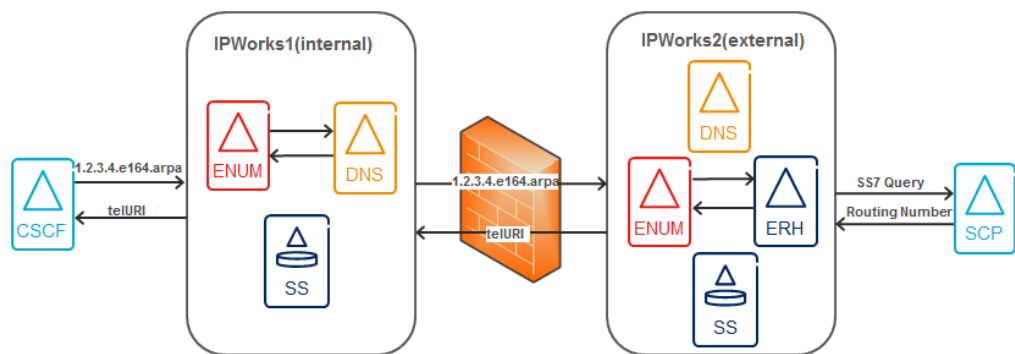


Figure 6 Split Namespace Architecture

Note: It is not recommended to configure IPWorks1 (internal) and IPWorks2 (external) in the same IPWorks cluster.

Table 4 Modules of IPWorks1 (internal) and IPWorks2 (external)

| Location | Module | Configuration for NP Query |
|----------------------|------------|--|
| DIPWorks1 (internal) | ENUM | Define the <i>EnumAcl</i> , <i>EnumView</i> and <i>EnumZVRel</i> ; Configure the attribute <i>destnode</i> of <i>EnumDnRange</i> to 5 in the internal <i>EnumView</i> to forward the query to BIND first. |
| | DNS (BIND) | Configure the <i>ForwardZones</i> to forward the NP queries to ENUM in IPWorks2 (external). The forwarding address needs to be specified in the option with the keyword "forwarders". |



| Location | Module | Configuration for NP Query |
|--------------------|------------|---|
| IPWorks (external) | ENUM | Define the <i>EnumAcl</i> , <i>EnumView</i> and <i>EnumZVRel</i> ; Configure the <i>EnumDnRange</i> with NP options. |
| | DNS (BIND) | Define the external <i>DnsServer</i> . |
| | ERH | Configure SS7 stack. |

The following example describes that an internal NP query is forwarded to SCP by IPWorks1 (internal) and IPWorks2 (external) in tandem.

For example, the environmental conditions are as follows:

- IP Address of DNS/ENUM server in IPWorks1 (internal) : 10.170.4.48
- IP Address of DNS/ENUM server in IPWorks2 (external) : 10.170.5.48
- Number Series Name: *.6.7.8.e164.arpa
- Subnet of Internal Network: 10.170.4.0/24

Check ENUM server information (see Section 4.1.2) , and configure the following objects in IPWorks1 (internal) and IPWorks2 (external) respectively.

On Storage Server of IPWorks1 (internal) :

1. Create an *EnumZone* object.

```
IPWorks> create enumzone 8 -set enumzonename=8.e164.arpa
```

2. Create an *EnumAcl* object.

```
IPWorks> create enumacl 1 -set aclname=Client;matchlist
="{10.170.4.0/24;}"
```

3. Create an *EnumView* object.

```
IPWorks> create EnumView 1 -set serveridlist=1;viewname
=internalView;aclid=1;rank=100
```

4. Create an *EnumZVRel* object.

```
IPWorks> create EnumZVRel -set zoneid=8;viewid=1
```

5. Create an *DnsServer* object.

```
IPWorks> create dnsserver idns -set dnsname=internal.com;ad
dress=10.170.4.48;option=
"listen-on port 5300 {any;}"
IPWorks> modify dnsserver idns -add option="allow-query-cache {any;}"
```



Note: If other areas exist, choose one of the areas for the `DnsServer` objects.

6. Create an *EnumDnRange* object.

```
IPWorks> create countrycode -set ccode=86
IPWorks> create destnode idnstest -set
type=5;
IPWorks> create enumdnrange 6.7.8.e164.arpa
-set viewid=1;destnode=idnstest
```

7. Create a *ForwardZone* object.

```
IPWorks> create forwardzone idns 8.e164.arpa -set
Option="forwarders {10.170.5.48;}", "forward only"
```

On Storage Server of IPWorks2 (external) :

1. Create an *EnumZone* object.

```
IPWorks> create enumzone 8 -set enumzonename=8.e164.arpa
```

2. Create an *EnumAcl* object.

```
IPWorks> create enumacl 2 -set aclname=Client;matchlist
="{10.170.4.48;!10.170.4.0/24;}"
```

3. Create an *EnumView* object.

```
IPWorks> create EnumView 2 -set serveridlist=2;viewname
=internalView;aclid=2;rank=200
```

4. Create an *EnumZVRel* object.

```
IPWorks> create EnumZVRel -set zoneid=8;viewid=2
```

5. Create a *DnsServer* object.

```
IPWorks> create dnsserver idns -set dnsname=exter
nal.com;address=10.170.4.48;option=
"listen-on port 5300 {any;}"
```

Note: If other areas exist, choose one of the areas for the `DnsServer` objects.

6. Create an *EnumDnRange* object with NP options.

```
IPWorks> create ainlnpdata -set
protocolsettingname=protocollnp
IPWorks> create ainnode -set
enumserverid=1;ainuserid=9999999999;localssn=100;localspc=
```



```
100;bearercapability=0;npswitch=1
IPWorks> create countrycode -set ccode=86
IPWorks> create destnode ednctest -set
type=7;protocolsettingname=protocolnp
IPWorks> create enumdnrange 6.7.8.e164.arpa
-set viewid=2;destnode=ednctest
```

4.7 Configuring ENUM Front End

ENUM as an FE interacts with BE-DB (CUDB) in UDC solution. ENUM FE supports:

1. User data (ENUMDN RANGE and ENUMDNSCHED) are stored in CUDB and cached in ENUM FE.
2. SOAP and LDAP are interfaces between ENUM FE and CUDB.
3. ENUM FE acts as the SOAP server (2N MODE) to handle SOAP notifications from CUDB when ENUNDN RANGE and ENUMDNSCHED are provisioning.

ENUM FE Sync is installed on PL-3 and PL-4, which has the following two functionalities:

4. When an SOAP notification arrives, ENUM FE Sync updates ENUMDNSCHED/ENUMDN RANGE in NDB.
5. When the timer of ENUMDNSCHED or ENUMDN RANGE is expired, ENUM FE Sync server deletes the expired ENUMDNSCHED, or refreshes (deletes all and fetches from CUDB) ENUMDN RANGE.

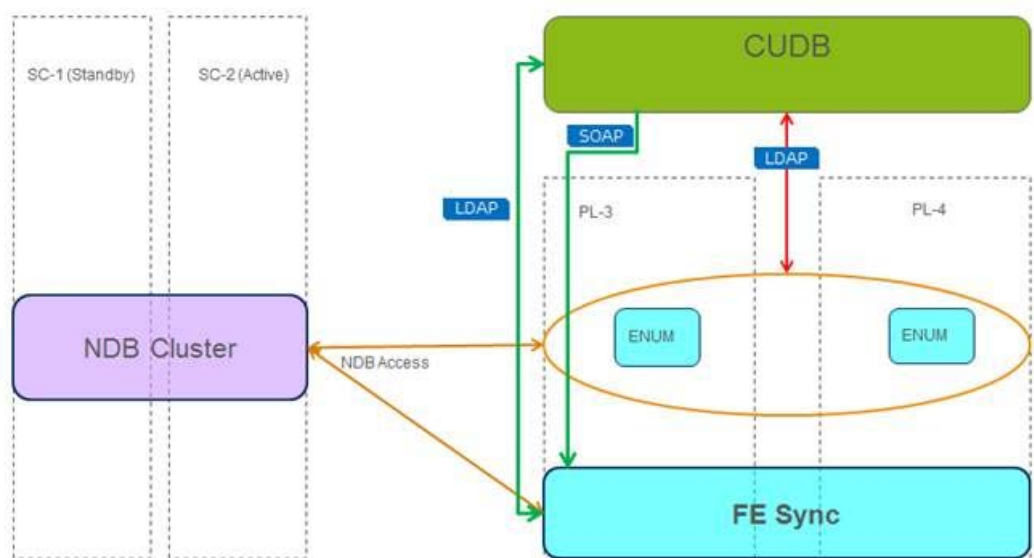


Figure 7 ENUM FE Architecture



The configuration of ENUM Front End contains the following topics:

- Section 4.7.1 Enabling ENUM Front End Feature on page 120
- Section 4.7.2 Configuring Cached EnumDnSched from CUDB in the Local on page 121
- Section 4.7.3 Configuring Cached EnumDnSched Expiration on page 121
- Section 4.7.4 Configuring Cached EnumDnRange Expiration on page 122
- Section 4.7.5 Configuring ENUM Response Policy when LDAP Fails on page 122
- Section 4.7.6 Configuring EnumZone according to CUDB ENUM records on page 123
- Section 4.7.7 Refreshing ENUMDNSHED and ENUMDNRRANGE in Cache Manually on page 123
- Section 4.7.8 Configuring CUDB Connection Pool on page 124
- Section 4.7.9 Configuring LDAP Dictionary on page 127
- Section 4.7.10 Checking Default SOAP Server on page 128
- Section 4.7.11 Configuring ENUM Front End Counters in CUDB on page 129

4.7.1 Enabling ENUM Front End Feature

To enable the ENUM Front End feature:

1. Log on to the ECLI.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Enable the ENUM Front-End feature by configuring the MO *EnumFE*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumFE=1
(config-EnumFE=1)> enableEnumFE=true
(config-EnumFE=1)>commit
(config-EnumFE=1)>exit
```

3. Restart the ENUM Server and ENUM FE Sync Server to let the changes take effect.

```
# ipw-ctr restart enum <PL hostname>
# ipw-ctr restart fesync <PL hostname>
```



4. Synchronize the ENUM Front-End configuration to Storage Server.

```
# ipw-ctr restart ss <SC hostname>
```

4.7.2 Configuring Cached EnumDnSched from CUDB in the Local

To configure the cached EnumDnSched from CUDB in the local:

1. Log on to the ECLI.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Configure the cached EnumDnSched from CUDB in the local by configuring the MO *EnumFE*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumFE=1
(config-EnumFE=1)>enableEnumDnSchedCache=true
(config-EnumFE=1)>commit
(config-EnumFE=1)>exit
```

Note: When `enableEnumDnSchedCache=false`, all the cached EnumDnSched in the local will be removed.

3. Restart the ENUM Server and ENUM FE Sync Server to let the changes take effect.

```
# ipw-ctr restart enum <PL hostname>
# ipw-ctr restart fesync <PL hostname>
```

4.7.3 Configuring Cached EnumDnSched Expiration

To configure cached EnumDnSched expiration:

1. Log on to the ECLI.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

2. Configure the cached EnumDnSched expiration by configuring the MO *EnumFE*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumFE=1
(config-EnumFE=1)>EnumDnSchedExpiration=7
(config-EnumFE=1)>commit
(config-EnumFE=1)>exit
```

Note: The cached EnumDnSched are deleted automatically per every EnumDnSchedExpiration (7 days by default).



- Restart the ENUM FE Sync Server to let the changes take effect.

```
# ipw-ctr restart fesync <PL hostname>
```

4.7.4 Configuring Cached EnumDnRange Expiration

To configure the cached EnumDnRange expiration:

- Log on to the ECLI.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

- Configure the cached EnumDnRange expiration by configuring the MO *EnumFE*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumFE=1
(config-EnumFE=1)>EnumDnRangeExpiration=7
(config-EnumFE=1)>commit
(config-EnumFE=1)>exit
```

Note: The cached EnumDnRange are deleted automatically per every EnumDnSchedExpiration (7 days by default).

- Restart the ENUM FE Sync Server to let the changes take effect.

```
# ipw-ctr restart fesync <PL hostname>
```

4.7.5 Configuring ENUM Response Policy when LDAP Fails

To configure ENUM response:

- Log on to the ECLI.

```
# ssh <username>@<MIP_OAM_IP> -t -s cli
```

- Configure the ENUM response policy when configuring the MO *EnumFE*.

```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumFE=1
(config-EnumFE=1)>HandleLDAPFailure=NXdomain
(config-EnumFE=1)>commit
(config-EnumFE=1)>exit
```



Note: NXdomain indicates how ENUM server responds to ENUM client when receiving a busy LDAP response or timeout. enumeration<NXdomain, Discard, Serverfail>, NXdomain is used by default.

3. Restart the ENUM Server to let the changes take effect.

```
# ipw-ctr restart enum <PL hostname>
```

4.7.6

Configuring EnumZone according to CUDB ENUM records

Each the ENUM record in CUDB must match an ENUM Zone, otherwise, it's unavailable to ENUM Server. For NAPTR queries, only those can find valid zones in ENUM Server will be continued with the following ENUM processing. For example, two NAPTRs in CUDB,

```
fqdn=1.2.3.4.5.6.7.8.9.0.3.3.1.e164.iptelco.com
fqdn=1.2.3.4.5.6.7.8.9.0.3.3.2.e164.iptelco.com
```

An EnumZone e164.iptelco.com object must be created:

```
IPWorks>create enumzone 1 -set enumzonename="e164.iptelco.com"
IPWorks>exit
```

Note: It will have performance impact to configure EnumZone when the ENUM FE is running.

4.7.7

Refreshing ENUMDNSHED and ENUMDNRRANGE in Cache Manually

1. Log on to ENUM Server(PL-3 or PL-4).

```
#ssh <username>@<MIP_OAM_IP>
# ssh PL-3
```

2. Check if the status of fesync is active.

```
PL-3:~ # ipw-ctr status fesync
```

```
fesync on PL-3 is running, working as an active node.
```

3. Refresh ENUMDNSCHED and ENUMDNRRANGE in the cache manually on the PL with active fesync:

Note: manual_refresh must be executed only on the PL with active fesync. Otherwise, it fails with "Failed to connect to <VIP_DATA_IP> port <PORT of SOAP Server>: Connection refused".



```
PL-3:~ # /opt/ipworks/enumfe/scripts/manual_refresh ENUMDnSched
% Total      % Received % Xferd  Average Speed   Time    Time       Time    Current
           Dload  Upload   Total   Spent    Left     Speed
100   834      0   239      0   595   5172  12876  --:--:--  --:--:--  --:--:--    0

enum dnsched refresh successfully

PL-3:~ # /opt/ipworks/enumfe/scripts/manual_refresh ENUMDnRange
% Total      % Received % Xferd  Average Speed   Time    Time       Time    Current
           Dload  Upload   Total   Spent    Left     Speed
100   835      0   239      0   596    294    735  --:--:--  --:--:--  --:--:--    0

enum dnrange refresh successfully
```

4.7.8 Configuring CUDB Connection Pool

This section guides how to configure CUDB connection pool for ENUM server and NP.

Prerequisite:

If route is required for the CUDB connection, follow the examples described in *Configure Route for IPWorks PL Node*.

Table 5 lists the presupposition values that are used as an example for the CUDB connection configuration. CUDB connection configuration varies based on the actual environment. Both IPv4 and IPv6 are supported to work with CUDB.

Table 5 Example: CUDB Node Parameters Values

| CUDB Site Name | CUDB Node Name ⁽¹⁾ | CUDB Node Parameters |
|----------------|-------------------------------|---|
| site1 | node1 | Address = "192.168.20.11" |
| | | distinguishedName = "" |
| | | Password = "" |
| | | poolSize = 16 |
| | | Port = 389 |
| | node2 | Address = "192.168.20.12" |
| | | distinguishedName = "cudbUser=ENUMUser,ou=admin,dc=ericsson,dc=com" |
| | | Password = "secret" |
| | | poolSize = 16 |
| | | Port = 389 |



| CUDB Site Name | CUDB Node Name ⁽¹⁾ | CUDB Node Parameters |
|----------------|-------------------------------|---------------------------|
| site2 | node1 | Address = "192.168.20.13" |
| | | distinguishedName = "" |
| | | Password = "" |
| | | poolSize = 16 |
| | | Port = 389 |
| | node2 | Address = "192.168.20.14" |
| | | distinguishedName = "" |
| | | Password = "" |
| | | poolSize = 16 |
| | | Port = 389 |
| site3 | node1 | Address = "192.168.20.15" |
| | | distinguishedName = "" |
| | | Password = "" |
| | | poolSize = 16 |
| | | Port = 389 |
| | node2 | Address = "192.168.20.16" |
| | | distinguishedName = "" |
| | | Password = "" |
| | | poolSize = 16 |
| | | Port = 389 |

(1) Only CUDB AD Node can be used here for connection.



```

>ManagedElement=<Node Name>,IpworksFunction=1,IpworksCommonRoot=1,
DataBaseInfo=1,CudbManager=1,CudbServiceSite=<Service Name>,CudbSiteManager=1
(CudbSiteManager=1)>configure
(config-CudbSiteManager=1)>CudbSite=site1
(config-CudbSite=site1)>CudbNode=node1
(config-CudbNode=node1)>address=192.168.20.11
(config-CudbNode=node1)>poolSize=16
(config-CudbNode=node1)>show -v
CudbNode=node1
  address="192.168.20.11"
  cudbNodeId="node1"
  distinguishedName= [] <empty>
  password= [] <empty>
  poolSize=16
  port=389 <default>
(config-CudbNode=node1)>up
(config-CudbSite=site1)>CudbNode=node2
(config-CudbNode=node2)>address=192.168.20.12
(config-CudbNode=node2)>distinguishedName="cudbUser=ENUMUser,ou=admin,dc=ericsson,dc=com"
(config-CudbNode=node2)>password="secret" cleartext
(config-CudbNode=node2)>poolSize=16
(config-CudbNode=node2)>show -v
CudbNode=node2
  address="192.168.20.12"
  cudbNodeId="node2"
  distinguishedName="cudbUser=ENUMUser,ou=admin,dc=ericsson,dc=com"
  password="1:k8d2jPCL2Qa76V1jmjN6+CLUQIbQreeg"
  poolSize=16
  port=389 <default>
(config-CudbNode=node2)>commit

```

Example 10 CUDB Connection Configuration

- `<Node Name>` represents the node name for IPWorks; `<Service Name>` represents the IPWorks service that sends requests to CUDB. The value can be ENUM, NP, or AAA.
- The username and password of the `cudbUser` are created by the CUDB. If they are used, make sure that they have been created in CUDB before you restart ENUM server. If not, the related configuration is unnecessary and keep it empty.
- For ENUM and NP, the value of `poolSize` must be 16, because it is an optimized number.

Follow the Example 10 to configure the other CUDB sites and CUDB nodes. The final results are as below:

```

>dn ManagedElement=<Node Name>,IpworksFunction=1,IpworksCommonRoot=1,
DataBaseInfo=1,CudbManager=1,CudbServiceSite=<Service Name>,CudbSiteManager=1
(CudbSiteManager=1)>show -v CudbSite=site1
CudbSite=site1
  cudbSiteId="site1"
  CudbNode=node1
  CudbNode=node2
(CudbSiteManager=1)>show -v CudbSite=site2
CudbSite=site2
  cudbSiteId="site2"
  CudbNode=node1
  CudbNode=node2
(CudbSiteManager=1)>show -v CudbSite=site3
CudbSite=site3
  cudbSiteId="site3"
  CudbNode=node1
  CudbNode=node2

```



Note: IPWorks support up to 400 LDAP connections towards all the CUDB nodes.

CUDB Site Priority:

In the scenario of CUDB connection pool with multiple sites, for example, three CUDB sites (`site1`, `site2`, and `site3`), the site priority is as below:

```
site1 > site2 > site3
```

Which means:

- ENUM-FE, AAA-FE, or NP connects to the nodes in `site1` by default.
- If `site1` is unreachable, then it connects to `site2`.
- If both `site1` and `site2` are unreachable, then it connects to `site3`.
- If `site1` has recovered, then it switches back to `site1`.

ENUM-FE, AAA-FE, or NP does not connect to a lower priority site if a higher priority site (like `site1`) is available or recovered.

CUDB Site Priority Strategy

The CUDB site with a lower string name has a higher priority.

String name `<X>` is lower than string name `<Y>` in the following cases:

- Both string names are compared character by character. The value of the first unmatched character in string name `<X>` is lower than the character in string name `<Y>`. For example, `site1 > site2`.
- All compared characters match but string name `<X>` is shorter. For example: `site > site1`

4.7.9 Configuring LDAP Dictionary

```
# vi /etc/ipworks/ldapschema/ldap_dictionary.xml
```

```
...
<service name="ENUM">
  <cudbRootEntry name="serv=enum,ou=servCommonData,dc=o,dc=com">
    <searchDn name="viewid=0,fqdn=%s,ou=EnumDnSched,"/>
    <entryList>
      <entry name="NAPTRRecord">
        <dn name="recordid=%s,fqdn=%s,ou=%s,"/>
        <attr name="recordid" alias="recordid"/>
        <attr name="order" alias="naptrorder"/>
        <attr name="preference" alias="naptrpreference"/>
        <attr name="service" alias="naptrservice"/>
        <attr name="flags" alias="naptrflags"/>
      </entry>
    </entryList>
  </cudbRootEntry>
</service>
```



```

        <attr name="txt" alias="naptrtxt"/>
        <attr name="viewid" alias="viewid"/>
        <attr name="ttl" alias="ttl"/>
    </entry>
</entryList>
<searchObject name="ENUMDNSCHED">
    <searchRequest baseDn="searchDn" searchScope="1" filter="
    <searchResult>
        <entry ref="NAPTRRecord"/>
    </searchResult>
</searchObject>
<searchObject name="ENUMDNRANGES">
    <searchRequest baseDn="ou=EnumDnRange,serv=enum,ou=servCo
    <searchResult>
        <entry ref="NAPTRRecord"/>
    </searchResult>
</searchObject>
</service>
...

```

Make sure that the content is aligned with customer LDAP server configuration: `cudbRootEntry`, `searchDn`, `dn`, and `baseDn`. The `baseDn` in `searchRequest` of `ENUMDNRANGES` is a string appending the name of `cudbRootEntry`. Take the above content as an example.

4.7.10 Checking Default SOAP Server

Checking the default SOAP server is to make sure that the value of `port` is identical with the value of `dest_port`. In normal condition, it is unnecessary to change the configuration of SOAP Server.

Do the following steps:

1. Open the file `evip.xml`.

```
# vi /cluster/storage/system/config/evip-apr9010467/evip.xml
```

2. Check the value of attributes `dest` and `dest_port`:

- `dest`: The VIP of SOAP Server. It identifies which IP address the SOAP notification will be sent to.
- `dest_port`: The port of SOAP Server listening. It identifies which port the SOAP notification will be handled by FE Sync.

For example, take the following XML file content as default:



```
<flow_policy name="soap-listener" address_family="ipv4" protocol="tcp" target_pool="DAT_
dest="<VIP_LDAP_IP>"dest_port="8080"/>
```

3. Save the file `evip.xml`.
4. Open the file `axis2.xml`.

```
# vi /opt/ipworks/enumfe/conf/axis2.xml
```

5. Check the value of attribute `port`.

- `port`: The port of SOAP Server listening. It identifies which port the SOAP notification will be handled by FE Sync.

For example, take the following XML file content as default:

```
<transportReceiver name="http"
    class="org.apache.axis2.transport.http.SimpleHTTPServer">
    <parameter name="port">8080/parameter>
</transportReceiver>
```

6. Save the file `axis2.xml`.

Note: If the value of `port` in file `axis2.xml` is inconsistent with the value of `dest_port` in file `evip.xml`, it is recommended to update the file `axis2.xml` based on the value in file `evip.xml`.

4.7.11

Configuring ENUM Front End Counters in CUDB

If needed, configure the counters `ENUMNAPTRCNT` and `ENUMFQDNCNT` in CUDB.

For more information about the counters and how to do the configuration, refer to *IPWorks Application Counters in CUDB*.

4.8

ENUM Server IPv6 Support

The IPv6 support of IPWorks ENUM server covers the following aspects:

- ENUM server accepts incoming queries on both IPv4 and IPv6 transport at the same time. This is the default behavior of ENUM server.
- ENUM server forwards queries which cannot be resolved to the configured DNS resolver through IPv4 transport. That is to say, the DNS server working together with ENUM server must be configured with IPv4 address. The two attributes `dnsResolverIPAddress` and `dnsResolverPort` of the object `EnumServer` are configured the same way as that for IPv4, which means the `dnsResolverIPAddress` is of IPv4 format.
- Whenever ENUM server forwards a query to DNS server, it forwards the original client source IP address information (IPv4 or IPv6 format).



- ENUM ACL can include the addresses and subnets for both IPv4 and IPv6. ENUM server is able to allow or block the client to access the specific ENUM views for both IPv4 and IPv6 user. The `matchlist` for `enumacl` can be configured with IPv6 using IPWorks CLI.

Example:

```
IPWorks> create enumacl 1 -set aclname=acl1;matchlist="{fe80::1aa9:5ff:fe18:316b/64;}"
```

For more information about how to configure ENUM ACL, see Section 4.2 on page 88.

- ERH supports NP query over IPv6 transport using SIGTRAN.

4.9 ENUM Operations

This section describes common operations instructions for the ENUM server.

4.9.1 Starting and Stopping ENUM Server

IPWorks provides mechanisms for controlling the ENUM server once it is installed and operating. Users can use the `ipw-ctr` command to start or stop the server directly from the system where the server is in operation.

The following example shows how to start and stop the ENUM server:

1. log on to SC-1 or SC-2.

```
# ssh <username>@<MIP_OAM_IP>
Password:<Password>
```

2. Start ENUM server.

```
# ipw-ctr start enum <PL hostname>
start enum ==> success.
```

3. Stop ENUM server.

```
# ipw-ctr stop enum <PL hostname>
stop enum ==> success.
```

4.9.2 Viewing Server Logs

The ENUM server allows the viewing of logs. Ensure that log is enabled, otherwise no log file will be created.

The ENUM server log is configured by changing the attribute `log` of the `MO EnumServer`. A CLI example to enable the log is shown in Example 11.



```
>configure
(config)>dn ManagedElement=<Node Name>,IpworksFunction=1,
IpworksDnsRoot=1,IpworksEnumRoot=1,EnumServer=1
(config-EnumServer=1)>Log=1
(config-log)>level=LOG_LEVEL_INFO
(config-log)>commit
```

Example 11 Enabling ENUM Server Log

The directory that stores all kinds of logs is /cluster/storage/no-backup/ipworks/logs. The ENUM related log file is /cluster/storage/no-backup/ipworks/logs/<hostname>/ipwenum.log. Where: The is <hostname> represents the PL hostname that holds the ENUM server. For example, PL-3. The user can see the ENUM server log by using the following command:

```
tail -5 /cluster/storage/no-backup/ipworks/logs/PL-3/ipwenum.log
```





Reference List

- [1] *Trademark Information*
- [2] *Typographic Conventions*
- [3] *Glossary of Terms and Acronyms*
- [4] *License Management*
- [5] *IPWorks Configuration Management*
- [6] *IPWorks DNS, ASDNS, ENUM Parameter Description*
- [7] *Managed Object Model (MOM)*
- [8] *Command Line Interface User Guide for IPWorks SS*
- [9] *Ericsson Command-Line Interface User Guide*
- [10] <http://www.ipv6.com/articles/general/ipv6-the-next-generation-internet.htm>
- [11] *IPWorks DNS Management User Guide*